IJCNLP 2008

Third International Joint Conference on Natural Language Processing

Proceedings of the Conference

Organizer

Asian Federation of Natural Language Processing

Local Host

International Institute of Information Technology, India

January 7-12 2008
Hyderabad, India

Hosts/Organizers

Asian Federation of Natural Language Processing (AFNLP)

International Institute of Information Technology, Hyderabad, India (IIIT-H)

Natural Language Processing Association of India (NLPAI)

Supporters

Yahoo! Research & Development, India

Department of Information Technology, MCIT, Government of India

Information & Communication Technology, Government of Andhra Pradesh

Microsoft Research, India

Tata Consultancy Services Limited

Centre for Development of Advanced Computing

Indian School of Business

Satyam Computer Service Limited

Google India

IBM India

Defense Research and Development Organization

Council for Scientific and Industrial Research

Vaakkriti: Sanskrit Tokenizer

Aasish Pappu and Ratna Sanyal
Indian Institute of Information Technology, Allahabad (U.P.), India
{ akpappu_b03, rsanyal}@iiita.ac.in

Abstract

Machine Translation has evolved tremendously in the recent time and stood as center of research interest for many computer scientists. Developing a Machine Translation system for ancient languages is much more fascinating and challenging task. A detailed study of Sanskrit language reveals that its well-structured and finely organized grammar has affinity for automated translation systems. This paper provides necessary analysis of Sanskrit Grammar in the perspective of Machine Translation and also provides one of the possible solution for Samaas Vigraha(Compound Dissolution).

Keywords: Machine Translation, Sanskrit, Natural Language Parser, Samaas Vigraha, Tokenization

1 Introduction

Sanskrit language and its grammar had exterted an emphatic impact on Computer Science and related research areas. It has resulted to put in extensive efforts in the field of Machine Translation(hereafter referred as MT). MT of Sanskrit is never an easy task, because of structural vastness of its Grammar. Besides, its strutural vastness Sanskrit Grammar is well organized and least ambigious compared to other natural languages, illustrated by the fact of increasing fascination for this ancient Aryan language. Its grammar possesses well organized rules and meta rules to infer those rules, thus proving to be a powerful analogy to context free grammar of a computer language.

Subsequently, it supports the idea of developing a parser for Sanskrit language, that would be helpful in developing a full-fledged MT system. As a part of development of parser, there are other important aspects to be taken care off. A *morphological analyser* and a *tokenizer* are two of the important components that play a vital role in the parser. A morphological analyser is used for identification of the base words from their morphonemes, further to understand the semantics of the original text. A tokenizer also plays its significant part in a parser, by identifying the group or collection of words, existing as a single and complex word in a sentence. Later on, it breaks up the complex word into its constituents in their appropriate forms. In Sanskrit, mainly we have two categories of complex words. They are

- Sandhi
- Samaas

1.1 Sandhi and Samaas

Sandhi: When two words combine to produce a new word whose point of combination is result of annihilation of case-end of former word and case-begin of latter. In short, the resulted new character that has been created at the point of combination is exactly equivalent to the sound produced when those two words are uttered without a pause. The inverse procedure to Sandhi-formation is known as *Sandhi Wicched*.

On the other hand, when two or more words are combined, based on their semantics then the resulting word is known as *Samaas* or Compound. Unlike

Sandhi, the point of combination in Samaas may or may not be a deformed in the resulting word. The inverse procedure of break-up of a Samaas is known as *Samaas Vigraha*. Considering the complexity of this problem, we restricted our focus to Samaas Vigraha or *Compound Dissolution*(hereafter Compound Dissolution is referred as CD for convenience).

1.2 Organization of the Paper

Initially, we would discuss about the problem of focus and the main objective of this paper in detail. Further, a little overview about the Sanskrit grammar and Knowledge Representation, that are required to understand the underlying concepts of the system. Then, we would brief about the existing systems in this areas and the related areas of interest. Later on, we would give a detailed description of the architecture of Vaakkriti. We would give a detailed analysis of the results of our system and finally, throw some light over our contribution to this research area.We shall conclude with some of drawbacks of our system and the challenges we have faced.

2 The Problem

Semantics being the prime focus, we need to learn the factors that effect the formation of a compound from the set of atomic words. The basic problem is identification of factors, by thorough analysis of language structure or with the help of a linguist. Especially various examples of Samaas must be extensively observed. After identification of factors, we need to find out the appropriate form of Knowledge Representation for the rule-base. Here, knowledge being the rules, based on which a particular compound is formed. The importance of CD can be clearly understood, during the process of tokenization. A well-defined set of rules in Sanskrit can be found in "Ashtadyayi", authored by 3^{rd} century grammarian and linguist Panini. Ashtadyayi contains rules of Grammar in a concise form, distributed over eight chapters. Our rule-base system would be based on the work of Kale et. al, that has detailed description of Paninian Grammar.

3 Sanskrit Grammar

As we have already mentioned that, it is necessary to know some of the basic concepts of the Sanskrit grammar. First, we would give some important definitions of terms that are frequently used in this paper.

3.1 Important Definitions

3.1.1 Vibhakti(Declension)

Sanskrit is a highly inflected language with three grammatical genders (masculine, feminine, neuter) and three numbers (singular, plural, dual). It has eight cases: nominative, vocative, accusative, instrumental, dative, ablative, genitive, and locative.

3.1.2 Dhatupata(Verbal Conjugation)

The verbs tenses (a very inexact application of the word, since more distinctions than simply tense are expressed) are organized into four 'systems' (as well as gerunds and infinitives, and such creatures as intensives or frequentatives, desideratives, causatives, and benedictives derived from more basic forms) based on the different stem forms (derived from verbal roots) used in conjugation. There are four tense systems:

- Present (Present, Imperfect, Imperative, Optative)
- Perfect
- Aorist
- Future (Future, Conditional)

3.2 Factors that effect

The list of factors that are involved in a rule are

- Part of Speech(hereafter referred as POS)
- List of Words(a token must be among a set of words to satisfy a rule)
- Case-End
- Case-Begin
- Declension
- Sense(a token with a particular sense is only qualified)
- Meaning
- Affix

- Affix Type(*Taddita* and *Kriti*)
- Number(sng, two, mny)(hereafter we refer number as num)
- Gender(mas, fem, neu)

The list of actions that act as functions in the consequent of a rule are:-

- setDecl(set the declension case for a specified token)
- addBefore(add a string before a specified token)
- addAfter(add a string after a specified token)
- setNumber(set the number of a token(sng,two,mny))
- replace(replace a token with a string related to it)

3.3 Compounds

Nominal compounds occur with various structures, however morphologically speaking they are essentially the same. Each noun (or adjective) is in its (weak) stem form, with only the final element receiving case inflection. Some examples of nominal compounds include:

Itaretara

Example: रामलक्ष्मणभरतशत्रुघ्ना:(RAmaLakshmaNaBaratAH) to रामः च लक्ष्मणः च भरतः च शत्रुघ्नः(RAma ca, LakshmaNa ca, Barata ca)

Rule: $\forall token$ POS($token$,noun) $\Rightarrow$ setDecl($token$,nom)$\wedge$ addAfter($token$,च)

Samahaara

Example: पाणीपादौ(pANIpAdau)to पाणी च पाददम् च(pANI ca pADam)

Rule: $\forall token, \exists sense$ POS($token$,noun) $\wedge$ SenseOf($token, sense$) $\Rightarrow$ setDecl($token$,nom)$\wedge$ addAfter($token$,च)

Dvitiya(Accusative) Tatpurusha

Example: दुःखातीतः(dukhatItaH)to दुःखम् अतीतः(dukham atItaH)

Rule: POS($token1$,noun) $\wedge$ WordList($token2$,स्रित, अतीत, पतित, गत, अत्यस्त, प्राप्त, आपन्न, गमी, बुभुक्षु) $\Rightarrow$setDecl($token1$,acc)

Trutiya(Instrumental) Tatpurusha

Example: दुःखातीतःto दुःखम् अतीतः

Rule: POS($token1$,noun) $\wedge$ (POS($token2$,verb) $\vee$ WordList($token2$,पूर्व , सदृश , ऊन))$\Rightarrow$ setDecl($token1$,ins)

Chaturthi(Dative) Tatpurusha

Example: यूपदारु(yupadaru)to यूपय दारु(yupaya daru)

Rule: POS($token1$,noun) $\wedge$ (Sense($token2$,"material") $\vee$ WordList($token2$,अर्थ , बलि , हित , सुख , रक्षित))$\Rightarrow$ setDecl($token1$,dat)

Panchami(Ablative) Tatpurusha

Example: चौरभयम्(cOrabayam)to चौराद् भयम्(cOraad bayam)

Rule: POS($token1$,noun) $\wedge$ (WordList($token2$, भय , भीत , भीति , भीः , अपेत , अपोढ , मुक्त , पतित , अपवस्त))$\Rightarrow$ setDecl($token1$,abl)

Shashti(Genitive) Tatpurusha

Example: राजपुरुषः(rAjapurushaH)to राजज्ञ पुरुषः(rAjangya PurushaH)

Rule: POS($token1$,noun) $\wedge$ (POS($token2$,noun)$\wedge\neg$ POS($token2$,verb)$\wedge\neg$ NumeralType($token2$,ordinal)$\wedge\neg$ SenseOf($token2$,"quality"))$\Rightarrow$ setDecl($token1$,gen)

Saptami(Locative) Tatpurusha

Example: नगरकाकः(nagarAkAkaH)to नगरे काकः इव(nagare kAkaH iva)

Rule: POS($token1$,noun) $\wedge$ (MeaningOf($token2$,"crow")$\wedge$ SenseOf($token2$,"contempt"))$\Rightarrow$ setDecl($token1$,loc)$\wedge$ addAfter($token2$, इव)

4 Knowledge Representation

We have already learnt that the process of CD is supported by a rule-base system. A production system is a good illustration to understand a rule-base system. To represent a complex rule, it would be better to use First Order Predicate Logic(FOPL). Under FOPL a rule can be written as of the form $P(a) \wedge Q(a) \wedge Q(b) \wedge R(c) \Rightarrow Action_1(a) \wedge Action_2(b) \wedge Action_1(c)$

where P, Q and R are predicates

a, b and c are constant symbols

$Action$ is a function symbol

The rule-base system of Vaakkriti is developed considering the factors as predicates and the tokens as constant symbols. A sample rule would look like this

$POS(tok1, noun) \wedge (POS(tok2, verb)\ decl(tok2, acc)) \Rightarrow setDecl(token1, acc)$.

5 Related Work

In the recent times many efforts have been made to develop various utilities for Sanskrit. The tools developed includes Sanskrit to other Indian Language transliteration tools, simple and primitive translation tools, many grammar analysing tools and many more learning tools. Some of the important works includes Anusaraka, a primitive machine translation tool developed by Akshar et. al. Anusaraka tries to take advantage of the relative strengths of the computer and the human reader, where the computer takes the language load and leaves the world knowledge load on the reader. Besides, these tools, there are some beautiful theory-based research work was also done. The concept of Indian Network Language(INL) is one of such concepts that was proposed by Anupam et. al. It gives a hypothesis to consider Sanskrit as INL because of its important properties like free word order and inherent semantic net structure. There are few other interesting research concepts that have been analysed in the context of Sanskrit language. Rick Braggs et. al have shown in his article how Knowledge Representation in the language of Sanskrit is one of those wonderful concept to show that Semantic Nets. Semantic Nets are concept respresenting structures, that show how a concept is related to other concepts semantically, a semantic net would like in the figure below. Another beautiful research work was comparison of Paninian Grammar and Computer language Grammar. Bhate et al. has analysed to show that how well organized and structured is Sanskrit Grammar and its forgotten valuable contributions to the field of Computer Science.

6 Architecture

An Itrans standard formatted devanagiri text is given as input to the system and the output of the system is the set of tokens produced after CD. The list of components in the system are listed below:

- Input Processor
- Symbol Table
- Knowledge Base
- Inference Engine
- Database
- Rule-Base Editor

The architecture of Vaakkriti can be seen in the figure

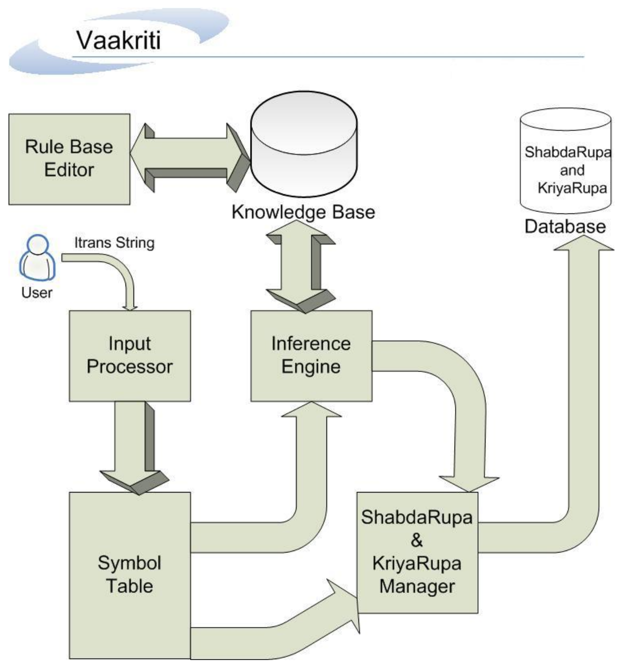


Figure 1: Architecture of Vaakriti

The algorithm of Vakkriti is given below:- A detailed description of each component is as follows.

Algorithm 1 Algorithm of Vaakkriti

input ← Itrans-Devanagiri Text
input′ ← breakUp(input)
tokenList ← tentativeTokenize(input′)
tokenInfoList ← tokenList
for token$_i$ in tokenInfoList **do**
    token$_(i)$ ← extractInfo(token$_i$
    $update$ token$_(i)$ in tokenInfoList
end for
for each rule(r) in Knowledge-Base(KB) **do**
    result ← infer(r,tokenInfoList)
    if result is $true$ **then**
        $return$ r
    end if
end for

6.1 Input Processor

The unstemmed compound taken as input to the system is a string in itrans format. First, Input Processor breaks the itrans string into chunks of characters on the basis of Devanagiri Character set. The heuristic for break up procedure is given below:-

The reason behind the breakup procedure is to ease the process of breaking the string into words in their tentative forms. If a string is considered as it is without breakup into devanagiri characters, then there is a high chance of ambiguity while lookup in the dictionary. For example:-

Without breakup of input string

```
aja
ajagaraH-- Found this word
```

With breakup of string into character sequences

```
a,ja
a,ja,ga,raH
```

Later on the chunks of characters are processed as in the procedure below:-

The words lying in input string are tentatively guessed by maintaining a stack of character sequences, thus checking with the dictionary for the right word. But, in most of the cases, the word in the input string do not have an exact match in the dictionary. This is because of the *matra* appended to *Case-End* of a word. Therefore, we have generated tokens for each matra and tried to find it in the dictionary. If the word is found, then the word along with its meaning is stored in the Symbol Table.

6.2 Symbol Table

Now, we shall discuss more about how a Symbol Table fetches those subtle information of a token. Symbol table extracts token information in the following manner:-

6.2.1 Part of Speech

Part of Speech is identified with the help of standard Monier Williams Dictionary, List of Adverbs, List of Prepositions, List of Numerals.

6.2.2 Sense and Meaning

First, meaning of the token is known from the dictionary and the sense of the token is fetched through a special kind of procedure. The technique has following steps:-

1. Identify the nouns in the meaning phrase.
2. Find sense for each noun with the help of English Wordnet.
3. Find a list of "common" senses for all the nouns.
4. That list of senses is assumed to the sense of a token.

6.2.3 Gender and Number

These are fetched from the XML database.

6.3 Knowledge Base

The Knowledge Base(KB) contains facts and rules that supports the system, for identifying a given input. The KB has been classified well, according to the Rule Sets. A Rule Set is a set of rules that are meant for a particular type of compound. Infact, a new rule set can be created whenever there is a new part of speech to be dealt with. It has been assumed that, a rule has clauses(both unit and definite) on antescendent side, whose number is equal to tentative number of tokens in the input parsed string. On the other hand, the consequent or conclusion contains the list of actions that has to be operated over the tokens(in the input string) by the system. More about the rule structure in the next section.

The KB is well integrated with the Rule Base Editor(RBE) and the Inference Engine. Currently, it contains limited number of rules this makes the KB non-monotonic, yet it can be made monotonic, by addition of new rules.

6.4 Database

There is a large database that supports the whole system of Vaakriti. The database is contained in the form of XML files. There are following tables in the database:-

- Nouns, Adjectives, Numerals Declensions.
- Adverbs, Conjunctions and Prepositions.
- Dictionary Database.
- Preverbs database.
- Other Morphonemes.

6.5 Inference Engine

Whenever premises of a particular are satisified by the input parse string, then it is said that a rule is fired. A fired rule applies its consequent part over the parsed string to result in actual goal. This procedure is known as Rule Inference.

6.6 Rule Base Editor

The sole motive of Rule-Base Editor is to free the Knowledge Engineer free from rule entry. A Linguist with little training to operate the GUI can be provided, would suffice this task.

7 Results

The system has been tested with many examples that have been taken from the book written by Kale et al. The set of examples have been chosen from different set of Compounds. In most of the cases system has given correct results with a precision of 90%, but in some of the cases that involve sense, it became quite difficult to produce the result. Lack of linguistic tools like Wordnet for Sanskrit language imposes limitations on word sense disambiguation. We have developed a sense list for a limited set of words by observing some of the important sanskrit texts, based on the knowledge we have acquired.

8 Our Contribution

We have proposed a utility called Rule-Base Editor, besides our actual work on CD. The motive behind Rule-Base Editor is to induce the property of flexibility into the system. It always avails a linguist to enter new rules with the help of Rule-Base Editor without any support from knowledge engineer.

We have already learnt that Samaas Vigraha(CD) is the most important aspect of the tokenization phase in a parser. Implicitly, the acquisition of factors and rules also gather equal importance. Signifying this fact, we have done rigorous survey over the grammar to identify these factors. Hence, we assert that our system will be a significant contribution in this area of research.

9 Future Scope and Conclusion

We assert that Vaakkriti would be a preliminary contribution to the realm of NLP. Adding to the major works that have been done already, Vaakkriti is an attempt to enhance the existing works. We would extend the current system and develop a full-fledged parser that will suffice most of the requirements of MTsystem.

Although, it looks the way that the problem has been solved, but the actual problems arouses when a Sanskrit poem is given as input to a MT system. Usually, a sanskrit poem conveys more than one meaning and sometimes figure of speech is used, that adds fuel to the fire. This becomes a herculean task for a MT system and it will remain as a myth forever.

Acknowledgements

The authors would like to specially thank Gerard Huet for providing linguistic database of declensions and verbal roots, that was quite helpful in making our system fine and complete. The authors gratefully acknowledge financial support from the Universal Digital Library project, funded by the Ministry of Communication and Information Technology (MCIT) India and also Indian Institute of Information Technology, Allahabad.

References

Higher Sanskrit Grammar, M. R. Kale, Motilal Banarasi-Dass Publishers.

"Paninis Grammar and Computer Science", Saroja Bhate and Subhash Kak, Annals of the Bhandarkar Oriental Research Institute, vol. 72, 1993, pp. 79-94.

"Knowledge Representation in Sanskrit and Artificial Intelligence", Rick Briggs

"Artificial Intelligence", Elain Rich and Kevin Knight, 2nd Edition, Tata McGrawHill, 1991.

"Artificial Intelligence, A Modern Approach" Stuart Russell and Peter Norvig, 2nd Edition, Pearson Education, 2003.

"Sanskrit as Indian Networking Language: A Sanskrit Parser", Anupam, 2004.

"Natural Language Processing, A Paninian Perspective", Akshar Bharti, Vineet Chaitanya and Rajeev Sangal, Pearson Education.

"Natural Language Processing using PROLOG" by M. Gerald, M Chris, Addison and Wisley, 1989.

"Cognitive Science Learning Resource", http://www.comp.leeds.ac.uk/ugadmit/cogsci/knowled

A Bottom up Approach to Persian Stemming

Amir Azim Sharifloo
NLP Research Lab,
Department of Electrical &
Computer Engineering,
Shahid Beheshti University,
Tehran, Iran
a.sharifloo@mail.sbu.ac.ir

Mehrnoush Shamsfard
NLP Research Lab,
Department of Electrical &
Computer Engineering,
Shahid Beheshti University,
Tehran, Iran
m-shams@sbu.ac.ir

Abstract

Stemmers have many applications in natural language processing and some fields such as information retrieval. Many algorithms have been proposed for stemming. In this paper, we propose a new algorithm for Persian language. Our algorithm is a bottom up algorithm that is capable to reorganize without changing the implementation. Our experiments show that the proposed algorithm has a suitable result in stemming and flexibility.

1 Introduction

In linguistics, *stem* is a form that unifies the elements in a set of morphologically similar words (Frakes and Yates, 1992), therefore *stemming* is the operation which determines the stem of a given word. In other words, the goal of a stemming algorithm is to reduce variant word forms to a common morphological root, called "stem" (Bacchin et al., 2002).

There are three common approaches that are used in stemming: affix stripping, lookup table and statistical methods (Bento et al., 2005). Affix stripping dependends on the morphological structure of the language. The stem is obtained by removing some morphemes from the one or both sides of the word. Porter algorithm (Porter, 1980) is an example of this kind of algorithms. This stemmer is made up of five steps, during which certain rules are applied to the words and the most common suffixes are removed.

In lookup table approach, each word and its related stem are stored in some kind of structured form. Consequently, for each stored word, we find its stem. However, the approach needs more space. Also, for each new word, table must be updated manually.

In statistical methods, through a process of inference and based on a corpus, rules are formulated regarding word formation. Some of the methodologies adopted are: frequency counts, n-gram (Mayfield and McNamee, 2003), link analysis (Bacchin et al., 2002), and Hidden Markov Models (Melucci and Orio, 2003). This approach does not require any linguistic knowledge whatsoever, being totally independent of the morphological structure of the target language.

In this paper, we propose a new algorithm for stemming in Persian. Our algorithm is rule based and in contrast with affix stripping approach, it is a stem based approach. That means, at first we find possible stems in the word, after that we check which stems are matched with rules.

Our algorithm is bottom up while affix stripping methods are top down. In other words, we try to generate the word using candidate stems of the word which we call *cores* of the word. If the word is generated, the stem is correct. On the other hand, affix stripping approaches try to removing affixes until reaching to any stem in the word.

Some stemming methods have been presented for Persian (Taghva et al., 2005) which use affix stripping approach. Our proposed method tries to reach better precision rather than previous methods. Also, this method tokenizes the word to morphemes which could employ in other morphological methods.

The paper is organized as follows: section 2 presents a brief review of Persian from morphological perspective; in section 3, we describe the proposed

algorithm in details; section 4 is about our experiments.

2 Persian from a Morphological Perspective

Persian is an Indo-European language, spoken and written primarily in Iran, Afghanistan, and a part of Tajikistan. It is written from right to left in the Arabic-like alphabet.

In Persian, verbs involve tense, number and person. For example[1], the verb “می خوانم” (*mi-xän-am:* I read) is a present tense verb consisting of three morphemes. “م” (*am*) is a suffix denoting first single person “خوان” (*xän*) is the present tense root of the verb and “می” (*mi*) is a prefix that expresses continuity.

If a verb has any object pronoun, it can be attached to the end of the verb such as “می‌خوانمش” (*mi-xän-am-aš*: I read it) in which “ش” (*aš*: it) is an object pronoun. Also, negative form of verbs is produced with adding “ن” (*ne*) to the first of them. For example, “نمی‌خوانم” (*ne-mi-xän-am* - I don’t read) is the negative form of the verb “می‌خوانم” (*mixänam* - I read). We have gathered 43 rules for verbs, some of them are shown in Table .1.

Table 1. Some rules for verbs in Persian

Rule	example
می + بن مضارع + شناسه مضارع (present person identifier + present root + mi)	می خوانم *(mi-xän-am)* (I read)
بن ماضی + ه + بود + شناسه ماضی (past person identifier + bud +eh + past root)	رفته بودم *(raft-e bud-am)* (I had gone)
ب + بن مضارع (present root + b)	بگذر *(be-gozar)* (Pass)
بن ماضی + ه + شد (shod + h + past root)	خوانده شد *(xand-e šod)* (it was read)

Nouns are more challengeable than others in Persian. We have gathered many rules for nouns that in following, we describe one of them. The plural forms of nouns are formed by adding the suffixes (ها, ان, ات, ون ,ین). “ها” (*hä*) is used for all words. “ان” (*än*) is used for humans, animals and every thing that is alive. Also, “ات, ون ,ین” (*ät ,un , in*) is used for some words borrowed from Arabic and some Persian words. We have another kind of plural form in Persian that is called *Mokassar* which is a derivational plural form (irregulars in Persian). Some examples of plural form are shown in Table 2.

Also, there are some orthographic rules which show the effects of joining affixes to the word. For example, consider that we have two parts of a word: A and B for joining as BA (Consider, Persian is written right to left). If the last letter of A and the first letter of B are “ا”(*ä*), one letter “ی” (*y*) is added between them. Assume A is “دانا” (*dänä* - wise) and B is “ان”(*än*), the joining result is “دانایان” (*dänä-yän:* wise people).

Table 2. Some kinds of plural form in Persian

Joining	*Result noun*
کشور + ها *(hä + kešvar)* (hä + country)	کشورها *(kešvar-hä)* (countries)
درخت + ان *(hä + deraxt)* (hä + tree)	درختان *(deraxt-än)* (trees)
(Mokassar form)کتب *(kotob)* (books)	کتب *(kotob)* (books)
آقا + ی + ان *(än + y + äghä)* (än + y + mister)	آقایان *(äghä-yän)* (men)

3 The Proposed Algorithm

Our algorithm is rule based and bottom up. At first, it tries to find substrings of the word that are stems or morphemes which are derived from any stem, we call them *cores*. After that, it joins each of cores with other elements of word for generating that word according to available rules. Finally, each core with at least one correct generation is a correct core and its stem is correct stem of the word. The algorithm includes three phases: 1. Substring tagging 2. Rule matching 3. Anti rule matching (Figure 1).

[1] Through the paper, we show Persian examples by their written form in Persian alphabet between “” followed by (*their pronunciation*: translation).

In substring tagging phase, we extract morphological information for all possible substrings of the word. At the end of this phase, we know which substrings of the word are morphemes and which ones are not. Also, we know clusters that each morpheme is their member. We use clusters for rule matching phase. Accordingly, we know cores in the word before beginning the second phase. We describe substring tagging details in section 3.1.

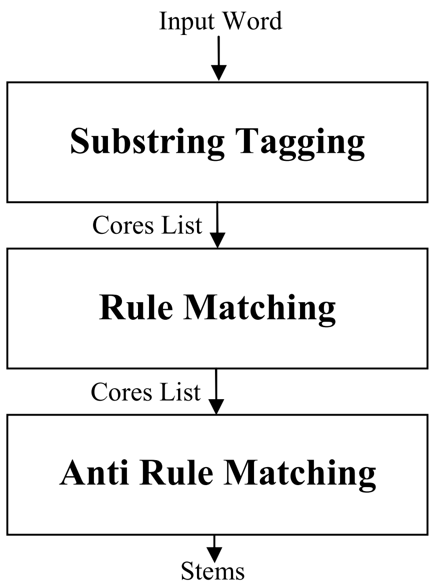


Figure1. Three phases of the proposed algorithm.

In rule matching phase, for each core that has been known in previous phase, we extract related rules. For example, “خوان” (*xän*) is one core of the word “میخوانم” (*mi-xän-am:* I read) and “بن مضارع” (*bone mozäre:* present root) is one of clusters that “خوان” (*xän)* is its member. Also, “م” (*am*) is a member of cluster “شناسه مضارع” (*šenase mozäre:* present person identifier) and “می” (*mi*) is a member of cluster “می” (*mi*). We have a rule in rules repository as:

(می + بن مضارع + شناسه مضارع)
(present person identifier + present root + mi)

where it is matched with the word ”میخوانم” (mi-xän-am: I read). Therefore, we find a matched rule for “خوان” (xän). At the end of second phase, each core that has extracted any possible rule for the word, remains in cores list and other cores are removed from it.

In anti-rule matching phase, we extract anti rules from anti rules repository for each core in the list. Each core which has any matched anti rule with the word morphemes, is removed from the cores list. At the end of the third phase, each stem of any core in the cores list is the correct stem for the word.

3.1 Substring Tagging

Every word with length N has N*(N+1)/2 substrings. Therefore, we need N*(N+1)/2 string matching for finding them in morphemes repository. We employ a Trie tree for storing morphemes and present an algorithm for retrieving morphological information from it that reduces the number of string matching. This algorithm needs N(N+1)/2 character matching (instead of string matching) at most. A simplified part of tree is shown in Figure 2.

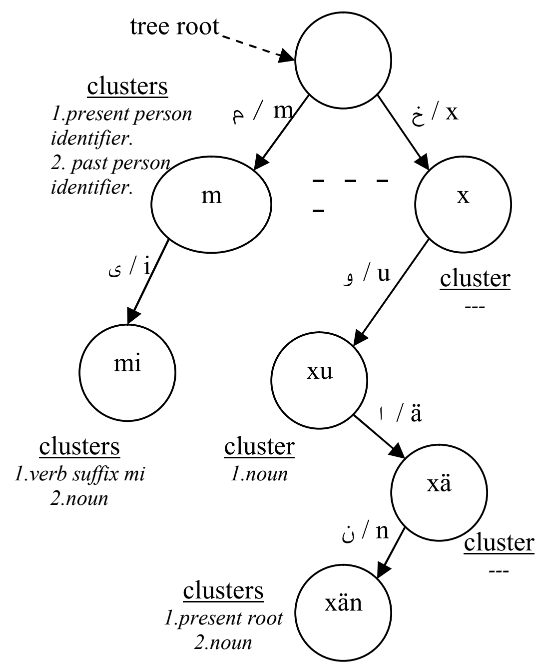


Figure 2. A simplified part of Trie tree that is used for storing morphological information.

The algorithm is described in the following:

We initiate N pointers (N is the word length) that they point to the tree root. Also, we use a counter C that is an index on the word. At first, C’s value is one that means its related letter is first letter of the word. At the end of each step, C is increased by one. Therefore, in each step, C points to one letter of the word that we call this letter L.

At first step, first pointer P1 finds one edge between root edges that its letter is equal with L. P1 goes down on tree by that edge. Here, P1 extract morphological information from its new position (a node of the tree) and fills morphological information for substring (1, 2).

At the second step, L is the second letter of the word, second pointer P2 finds one edge between root edges that its letter is equal with L. P2 goes down on tree by that edge, extract morphological information from its new position (a node of the tree) and fills morphological information for substring (2, 3). Also, P1 goes down on tree by an edge contained L, from its position that it is one of root children and fills morphological information for substring (1, 3).At third step, L is third letter of the word. Third pointer P3 starts from root and goes down on tree by an edge that its letter is equal with L and fills morphological information for substring (3, 4). P1, P2 repeat this work from their positions and fill morphological information for substring (1, 4) and (2, 4) respectively.

Next steps are done like these steps. Finally, we have obtained morphological information for all substrings of the word. Also, if one pointer could not find any edge with value L, it is blocked until the end of algorithm. Figure 3 shows pseudo code of this algorithm.

```
 Word: string;
P: array of pointer with word.length size;

for C = 1  to  word.length do
{
   for i = 1  to  C do
   {
       If (P[i] <> Blocked)
       {
         edge = find_edge( P[i] , L );
         // find_edge finds a edge from its position
         // in tree that its letter is equal with L.
         if ( edge <> null )
         {
           GoDown(P[i],edge);
           substring(i, C).mInfo = P[i]-> mInfo;
           // mInfo is morphological Information
         }
          else P[i] = Blocked;
      }
   }
}
```

Figure 3. The used algorithm for obtaining morphological information from Trie tree.

3.2 Rule Matching

We use many rules to generate correct words by morphemes in Persian. We store these rules in rules repository. Some gathered rules are shown in Table 3.

Table 3. Some gathered rules that we use.

Rule
ماضی ساده →بن ماضی + شناسه ماضی (past person identifier + past root → simple past)
امر →ب + بن مضارع (present root + b → imperative)
اسم جمع →اسم + ها (hä + noun → plural noun)
اسم جاندار جمع →اسم جاندار + ان (än + alive noun → alive plural noun)

Each rule is a sequence of clusters. A cluster represents a set of morphemes that affects role of them in the word. In other words, each morpheme could be applied as one or more roles for generating words. So, each role can be a cluster membership. For example, in English, “book” is a verb and a noun. But, As a noun, it has a plural form (books) and as a verb, it has a past form (booked).

Similarly, in Persian, the word “مرد” (mord: die) is a verb root and “مردند” (mord-and: They died) is a verb, too. Also, “مرد” (mard: man) is a noun and “مردها” (mard-hä: men) is one of its plural forms. In consequence, we put “مرد” in both of cluster “اسم” (esm: noun) and “بن ماضی” (bone mäzi: past root). We create a new cluster when a rule needs it and that cluster is not in clusters repository.

As we discussed about it, in Persian, we have several suffixes for plural form that every one is used for a set of nouns. The suffix “ها” (*hä*) is used for every noun and the suffix “ان” (*än*) is special for everything that is alive. Other suffixes are ap-

plied for some words borrowed from Arabic and some Persian words. A noun such as "پسر" (*pesar:* boy) has several plural forms (e.g. "پسرها"/*pesar-hä*, "پسران"/*pesar-än*). Therefore, we employ clusters for organizing this situation. For example, we put the morpheme "پسر" (*pesar:* boy) in cluster "اسم" (*esm:* noun) and "جاندار" (*jändär:* alive). Also, we have two rules in rules repository:

"اسم" + "ها"
(hä + noun)

and

"جاندار" + "ان"
(än + alive)

The morpheme "پسر" (*pesar:* boy) is a member of both clusters "اسم" (*esm:* noun) and "جاندار" (*jändär:* alive). Accordingly, these words "پسرها" (*pesar-hä:* boys) and "پسران" (*pesar-än:* boys) are correct form and their stem is "پسر" (*pesar:* boy). But about the morpheme "کتاب" (*ketäb:* book), it is a noun and a member of cluster "اسم" (*esm:* noun) but it is not a member of cluster "جاندار" (*jändär:* alive). So, "کتابها" (*ketäb-hä:* books) is a correct form and its stem is "کتاب" (*ketäb:* book). In contrast, "کتابان" (*ketäb-än*) is a wrong form and "کتاب" (*ketäb:* book) is not its stem. Also, we organize suffixes in similar cluster such as cluster "شناسه مضارع" (*šenase mozäre:* present person identifier), "حرف نفی فعل" (*harfe nafye fel*). Table 4 shows some clusters that we use.

Table 4. Some clusters that we use.

Cluster	Cluster
شناسه مضارع (present person identifier)	بن مضارع (present root)
پسوند جمع ها (plural suffix hä)	بن ماضی (past root)
پسوند جمع ان (plural suffix än)	اسم (noun)

At the end of this phase, each core must have a rule that it can generate the word. Otherwise, it is removed from cores list.

3.3 Anti Rule Matching

This phase is similar previous phase with a small difference. Like previous phase, we have a rules repository, but these rules are not applied in Persian. In fact, these rules are exceptions of previous phase rules. For example, we have a rule in rules repository:

(اسم جاندار + ان)
(än + alive noun)

On the other hand, there is an exception for this rule. Every noun with the final letter "ه" (he) can not use this rule. For example, "پرنده" (parand-e: bird) is a kind of animals with the final letter "ه" (he) and the word "پرندهان" (parand-e-än) is a wrong word in Persian. We call these exceptional rules "Anti rules".

The details of this phase: Each core from cores list retrieves the anti rules that they involve it. After that, each retrieved anti rule is checked with the morphemes in the word for possibility of word generation. Until now, all things were similar previous phase, but the difference is here. If there is any anti rule related to a rule of any core, that rule is removed from candidate rule list of that core. At the end of this phase, each core must have at least one rule that it can generate the word. Otherwise, it is removed from cores list. Finally, remained cores in cores list have correct stems of the word.

We have gathered a set of anti rules in a repository that each anti rule is related to a rule in rule repository. Some of these anti rules are shown in Table 5.

Table 5. Some gathered anti rules that we use.

Anti Rule
(اسم جاندارمنتهی به ه + ان) (an + alive noun ended with h)
(اسم منتهی به ا، و، ه، ی + ات) (at + noun ended with *ä,u,h, y*)

4 Experiments and Results

The most primitive method for assessing the performance of a stemmer is to examine its behavior when applied to samples of words - especially words which have already been arranged into 'conflation groups'. This way, specific errors (e.g., fail-

ing to merge "maintained" with "maintenance", or wrongly merging "experiment" with "experience") can be identified, and the rules adjusted accordingly.

We evaluated the proposed algorithm with a limited corpus of *Hamshahri* newspaper. We started with 252 rules and 20 anti rules. The algorithm retrieved 90.1 % of word stems correctly. The failed words are related to absence of some rules in rule repository or stems in Trie tree. Some of words in the corpus are shown in Table 6.

Table 6. Some of words in *Hamshahri* newspaper corpus

Stem	Word
ماجرا *(mäjarä)* (event)	ماجرای *(mäjarä-ye)* (event of)
رسم *(rasm)* (custom)	رسمها *(rasm-hä)* (customs)
پدیده *(padide)* (phenomenon)	پدیده های *(padid-e-hä-ye)* (phenomenons of)
بودن *(bud)* (to be)	بودند *(bud-and)* (They were)
ساعت *(säat)* (watch)	ساعتهایشان *(säat-hä-ye-šän)* (watch)
کشیدن *(kešidan)* (to draw)	بکشند *(be-keša-and)*
آخر *(äxar)* (end)	آخرین *(äxar-in)* (last)
رفتن *(raftan)* (going)	نرفته بودند *(na-raft-e budand)* (They had not gone)
سال *(säl)* (year)	امسال *(em-säl)* (this year)
مطالعه *(motäle'e)* (study)	مطالعات *(motäle-at)* (studies)
منطقه *(mantaghe)* (area)	مناطق *(manätegh)* (areas)

One of words could not be handle with our algorithm is “جابجا” (*jä-be-jä* - exchange). We discovered related rule for that and added it to rules repository. Therefore, if we evaluate the algorithm, the result will be better. Rules repository evolves and the algorithm result will be better without any change of program and code compilation.

5 Conclusion

In this paper, we proposed a bottom up method to stem Persian words. The main purpose of this method is high precision stemming based on morphological rules. The experiments show that it has suitable results in stemming and presents possibility of evolution easily.

References

Bento, Cardoso and Dias. 2005. *Progress in Artificial Intellegence*, 12th Portuguese Conference on Artificial Intelligence, pages 693 – 701.

Chris Paice. 1996. *Method for Evaluation of Stemming Algorithms Based on Error Counting*. JASIS , pages 632-649.

Frakes and Yates. 1992. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, NJ.

Mayfield and McNamee. 2003. *Single N-gram Stemming*. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information. retrieval, pages 415-416.

Melucci and Orio. 2003. *A Novel Method for Stemmer Generation Based on Hidden Markov Models*. In Proceedings of Conference on Information and Knowledge Management (CIKM03), pages 131-138.

Michela Bacchin, Nicola Ferro, and Massimo Melucci. 2002. *Experiments to evaluate a statistical stemming algorithm*. Working Notes for CLEF 2002, pages 161-168.

Michela Bacchin, Nicola Ferro, and Massimo Melucci. 2002. *The Effectiveness of a Graph-Based Algorithm for Stemming*. Springer-Verlag Berlin Heidelberg. pages 117–128.

Porter. *An Algorithm for Suffix Stripping*. 1980. Program. pages 130-137.

Taghva, Beckley and Sadeh. 2005. *A stemming algorithm for the Farsi language*. IEEE ITCC 2005, pages 158 - 162.

Named Entity Recognition in Bengali: A Conditional Random Field Approach

Asif Ekbal
Department of CSE
Jadavpur University
Kolkata-700032, India
asif.ekbal@gmail.com

Rejwanul Haque
Department of CSE
Jadavpur University
Kolkata-700032, India
rejwanul@gmail.com

Sivaji Bandyopadhyay
Department of CSE
Jadavpur University
Kolkata-700032, India
sivaji_cse_ju@yahoo.com

Abstract

This paper reports about the development of a Named Entity Recognition (NER) system for Bengali using the statistical Conditional Random Fields (CRFs). The system makes use of the different contextual information of the words along with the variety of features that are helpful in predicting the various named entity (NE) classes. A portion of the partially NE tagged Bengali news corpus, developed from the archive of a leading Bengali newspaper available in the web, has been used to develop the system. The training set consists of 150K words and has been manually annotated with a NE tagset of seventeen tags. Experimental results of the 10-fold cross validation test show the effectiveness of the proposed CRF based NER system with an overall average Recall, Precision and F-Score values of 93.8%, 87.8% and 90.7%, respectively.

1 Introduction

Named Entity Recognition (NER) is an important tool in almost all Natural Language Processing (NLP) application areas. Proper identification and classification of named entities (NEs) are very crucial and pose a very big challenge to the NLP researchers. The level of ambiguity in NER makes it difficult to attain human performance. NER has applications in several domains including information extraction, information retrieval, question-answering, automatic summarization, machine translation etc.

The current trend in NER is to use the machine-learning approach, which is more attractive in that it is trainable and adoptable and the maintenance of a machine-learning system is much cheaper than that of a rule-based one. The representative machine-learning approaches used in NER are Hidden Markov Model (HMM) (BBN's IdentiFinder in (Bikel et al., 1999)), Maximum Entropy (New York University's MENE in (Borthwick, 1999)) and Conditional Random Fields (CRFs) (Lafferty et al., 2001; McCallum and Li, 2003).

There is no concept of capitalization in Indian languages (ILs) like English and this fact makes the NER task more difficult and challenging in ILs. There has been very little work in the area of NER in ILs. In Indian languages particularly in Bengali, the work in NER can be found in (Ekbal and Bandyopadhyay, 2007a; Ekbal and Bandyopadhyay, 2007b) with pattern directed shallow parsing approach and in (Ekbal et al., 2007c) with HMM. Other than Bengali, a CRF based NER system can be found in (Li and McCallum, 2004) for Hindi.

2 Conditional Random Fields

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are used to calculate the conditional probability of values on designated output nodes given values on other designated input nodes. The conditional probability of a state sequence $S =< s_1, s_2, \ldots, s_T >$ given an observation sequence $O =< o_1, o_2, \ldots, o_T >$ is calculated as:

$$P_\wedge(s|o) = \frac{1}{Z_0} \exp(\sum_{t=1}^{T} \sum_{k} \lambda_k \times f_k(s_{t-1}, s_t, o, t)),$$

where, $f_k(s_{t-1}, s_t, o, t)$ is a feature function whose weight λ_k, is to be learned via training. The values of the feature functions may range between $-\infty, \ldots +\infty$, but typically they are binary. To make all conditional probabilities sum up to 1, we must calculate the normalization factor,

$$Z_0 = \sum_s \exp(\sum_{t=1}^{T} \sum_k \lambda_k \times f_k(s_{t-1}, s_t, o, t)),$$

which as in HMMs, can be obtained efficiently by dynamic programming.

To train a CRF, the objective function to be maximized is the penalized log-likelihood of the state sequences given the observation sequences:

$$L_\wedge = \sum_{i=1}^{N} \log(P_\wedge(s^{(i)}|o^{(i)})) - \sum_k \frac{\lambda_k^2}{2\sigma^2},$$

where $\{< o^{(i)}, s^{(i)} >\}$ is the labeled training data. The second sum corresponds to a zero-mean, σ^2 -variance Gaussian prior over parameters, which facilitates optimization by making the likelihood surface strictly convex. Here, we set parameters λ to maximize the penalized log-likelihood using Limited-memory BFGS (Sha and Pereira, 2003), a quasi-Newton method that is significantly more efficient, and which results in only minor changes in accuracy due to changes in λ.

When applying CRFs to the NER problem, an observation sequence is a token of a sentence or document of text and the state sequence is its corresponding label sequence. While CRFs generally can use real-valued functions, in our experiments maximum of the features are binary valued. A feature function $f_k(s_{t-1}, s_t, o, t)$ has a value of 0 for most cases and is only set to be 1, when s_{t-1}, s_t are certain states and the observation has certain properties. We have used the C++ based OpenNLP CRF++ package [1].

3 Named Entity Recognition in Bengali

Bengali is one of the widely used languages all over the world. It is the seventh popular language in the world, second in India and the national language of Bangladesh. A partially NE tagged Bengali news corpus (Ekbal and Bandyopadhyay, 2007d), developed from the archive of a widely read Bengali news paper available in the web, has been used in this work to identify and classify NEs. The corpus contains around 34 million word forms in ISCII (Indian Script Code for Information Interchange) and UTF-8 format. The *location*, *reporter*, *agency* and different *date* tags (*date*, *ed*, *bd*, *day*) in the partially NE tagged corpus help to identify some of the location, person, organization and miscellaneous names, respectively, that appear in some fixed places of the newspaper. These tags cannot detect the NEs within the actual news body. The date information obtained from the news corpus provides example of miscellaneous names. A portion of this partially NE tagged corpus has been manually annotated with the seventeen tags as described in Table 1.

NE tag	Meaning	Example
PER	Single-word person name	*sachin*/ PER
LOC	Single-word location name	*jadavpur*/LOC
ORG	Single-word organization name	*infosys*/ ORG
MISC	Single-word miscellaneous name	100%/ MISC
B-PER I-PER E-PER	Beginning, Internal or End of a multiword person name	*sachin*/B-PER *ramesh*/I-PER *tendulkar*/E-PER
B-LOC I-LOC E-LOC	Beginning, Internal or End of a multiword location name	*mahatma*/B-LOC *gandhi*/I-LOC *road*/E-LOC
B-ORG I-ORG E-ORG	Beginning, Internal or End of a multiword organization name	*bhaba*/B-ORG *atomic*/I-ORG *research*/I-ORG *center*/E-ORG
B-MISC I-MISC E-MISC	Beginning, Internal or End of a multiword miscellaneous name	*10e*/B-MISC *magh*/ I-MISC *1402*/E-MISC
NNE	Words that are not NEs	*neta*/NNE

Table 1: Named Entity Tagset

3.1 Named Entity Tagset

A CRF based NER system has been developed in this work to identify NEs in Bengali and classify them into the predefined four major categories, namely, 'Person name', 'Location name', 'Organization name' and 'Miscellaneous name'. In order to

[1] http://crfpp.sourceforge.net

properly denote the boundaries of NEs and to apply CRF in NER task, sixteen NE and one non-NE tags have been defined as shown in Table 1. In the output, sixteen NE tags are replaced appropriately with the four major NE tags by some simple heuristics.

3.2 Named Entity Features

Feature selection plays a crucial role in CRF framework. Experiments were carried out to find out the most suitable features for NER in Bengali. The main features for the NER task have been identified based on the different possible combination of available word and tag context. The features also include prefix and suffix for all words. The term prefix/suffix is a sequence of first/last few characters of a word, which may not be a linguistically meaningful prefix/suffix. The use of prefix/suffix information works well for highly inflected languages like the Indian languages. In addition, various gazetteer lists have been developed for use in the NER task. We have considered different combination from the following set for inspecting the best feature set for NER task: F=$\{w_{i-m}, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots w_{i+n}, |\text{prefix}| \leq n, |\text{suffix}| \leq n$, previous NE tag, POS tags, First word, Digit information, Gazetteer lists$\}$.

Following are the details of the set of features that were applied to the NER task:

• Context word feature: Previous and next words of a particular word might be used as a feature.

• Word suffix: Word suffix information is helpful to identify NEs. This feature can be used in two different ways. The first and the naïve one is, a fixed length word suffix of the current and/or the surrounding word(s) might be treated as feature. The second and the more helpful approach is to modify the feature as binary valued. Variable length suffixes of a word can be matched with predefined lists of useful suffixes for different classes of NEs. The different suffixes that may be particularly helpful in detecting person (e.g., *-babu*, *-da*, *-di* etc.) and location names (e.g., *-land*, *-pur*, *-lia* etc.) have been considered also. Here, both types of suffixes have been used.

• Word prefix: Prefix information of a word is also helpful. A fixed length prefix of the current and/or the surrounding word(s) might be treated as features.

• Part of Speech (POS) Information: The POS of the current and/or the surrounding word(s) can be used as features. Multiple POS information of the words can be a feature but it has not been used in the present work. The alternative and the better way is to use a coarse-grained POS tagger.

Here, we have used a CRF-based POS tagger, which was originally developed with the help of 26 different POS tags[2], defined for Indian languages. For NER, we have considered a coarse-grained POS tagger that has only the following POS tags:

NNC (Compound common noun), NN (Common noun), NNPC (Compound proper noun), NNP (Proper noun), PREP (Postpositions), QFNUM (Number quantifier) and Other (Other than the above).

The POS tagger is further modified with two POS tags (Nominal and Other) for incorporating the nominal POS information. Now, a binary valued feature 'nominalPOS' is defined as: If the current/previous/next word is 'Nominal' then the 'nominalPOS' feature of the corresponding word is set to 1; otherwise, it is set to 0. This 'nominalPOS' feature has been used additionally with the 7-tag POS feature. Sometimes, postpositions play an important role in NER as postpositions occur very frequently after a NE. A binary valued feature 'nominalPREP' is defined as: If the current word is nominal and the next word is PREP then the feature 'nomianlPREP' of the current word is set to 1, otherwise set to 0.

• Named Entity Information: The NE tag of the previous word is also considered as the feature. This is the only dynamic feature in the experiment.

• First word: If the current token is the first word of a sentence, then the feature 'FirstWord' is set to 1. Otherwise, it is set to 0.

• Digit features: Several binary digit features have been considered depending upon the presence and/or the number of digits in a token (e.g., ContainsDigit [token contains digits], FourDigit [token consists of four digits], TwoDigit [token consists of two digits]), combination of digits and punctuation symbols (e.g., ContainsDigitAndComma [token consists of digits and comma], ConatainsDigitAndPeriod [token consists of digits and periods]), combination of digits and symbols (e.g., ContainsDigitAndSlash [token consists of digit and slash],

[2]http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf

ContainsDigitAndHyphen [token consists of digits and hyphen], ContainsDigitAndPercentage [token consists of digits and percentages]). These binary valued features are helpful in recognizing miscellaneous NEs such as time expressions, monetary expressions, date expressions, percentages, numerical numbers etc.

• Gazetteer Lists: Various gazetteer lists have been developed from the partially NE tagged Bengali news corpus (Ekbal and Bandyopadhyay, 2007d). These lists have been used as the binary valued features of the CRF. If the current token is in a particular list then the corresponding feature is set to 1 for the current and/or the surrounding word(s); otherwise, set to 0. The following is the list of gazetteers:
(i) Organization suffix word (94 entries): This list contains the words that are helpful in identifying organization names (e.g., *kong*, *limited* etc). The feature 'OrganizationSuffix' is set to 1 for the current and the previous words.
(ii) Person prefix word (245 entries): This is useful for detecting person names (e.g., *sriman*, *sree*, *srimati* etc.). The feature 'PersonPrefix' is set to 1 for the current and the next two words.
(iii) Middle name (1,491 entries): These words generally appear inside the person names (e.g., *chandra*, *nath* etc.). The feature 'MiddleName' is set to 1 for the current, previous and the next words.
(iv) Surname (5,288 entries): These words usually appear at the end of person names as their parts. The feature 'SurName' is set to 1 for the current word.
(v) Common location word (547 entries): This list contains the words that are part of location names and appear at the end (e.g., *sarani*, *road*, *lane* etc.). The feature 'CommonLocation' is set to 1 for the current word.
(vi) Action verb (221 entries): A set of action verbs like *balen*, *ballen*, *ballo*, *shunllo*, *haslo* etc. often determines the presence of person names. The feature 'ActionVerb' is set to 1 for the previous word.
(vii) Frequent word (31,000 entries): A list of most frequently occurring words in the Bengali news corpus has been prepared using a part of the corpus. The feature 'RareWord' is set to 1 for those words that are not in this list.
(viii) Function words (743 entries): A list of function words has been prepared manually. The feature 'NonFunctionWord' is set to 1 for those words that are not in this list.
(ix) Designation words (947 entries): A list of common designation words has been prepared. This helps to identify the position of the NEs, particularly person names (e.g., *neta*, *sangsad*, *kheloar* etc.). The feature 'DesignationWord' is set to 1 for the next word.
(x) Person name (72, 206 entries): This list contains the first name of person names. The feature 'PersonName' is set to 1 for the current word.
(xi) Location name (7,870 entries): This list contains the location names and the feature 'LocationName' is set to 1 for the current word.
(xii) Organization name (2,225 entries): This list contains the organization names and the feature 'OrganizationName' is set to 1 for the current word.
(xiii) Month name (24 entries): This contains the name of all the twelve different months of both English and Bengali calendars. The feature 'MonthName' is set to 1 for the current word.
(xiv) Weekdays (14 entries): It contains the name of seven weekdays in Bengali and English both. The feature 'WeekDay' is set to 1 for the current word.

4 Experimental Results

A partially NE tagged Bengali news corpus (Ekbal and Bandyopadhyay, 2007d) has been used to create the training set for the NER experiment. Out of 34 million wordforms, a set of 150K wordforms has been manually annotated with the 17 tags as shown in Table 1 with the help of *Sanchay Editor* [3], a text editor for Indian languages. Around 20K NE tagged corpus has been selected as the development set and the rest 130K wordforms has been used as the training set of the CRF based NER system.

We define the *baseline* model as the one where the NE tag probabilities depend only on the current word: $P(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n) = \prod_{i=1,\ldots,n} P(t_i, w_i)$.

In this model, each word in the test data will be assigned the NE tag which occurred most frequently for that word in the training data. The unknown word is assigned the NE tag with the help of various gazetteers and NE suffix lists.

Ninety-five different experiments were conducted taking the different combinations from the set 'F' to

[3] Sourceforge.net/project/nlp-sanchay

Feature (word, tag)	FS (in %)
pw, cw, nw, FirstWord	71.31
pw2, pw, cw, nw, nw2, FirstWord	72.23
pw3, pw2, pw, cw, nw, nw2, nw3, FirstWord	71.12
pw2, pw, cw, nw, nw2, FirstWord, pt	74.91
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$pre$\rvert \leq 4$, $\lvert$suf$\rvert \leq 4$	77.61
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$, $\lvert$pre$\rvert \leq 3$	79.70
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$, $\lvert$pre$\rvert \leq 3$, Digit features	81.50
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$, $\lvert$pre$\rvert \leq 3$, Digit features, pp, cp, np	83.60
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$, $\lvert$pre$\rvert \leq 3$, Digit features, pp2, pp, cp, np, np2	82.20
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$, $\lvert$pre$\rvert \leq 3$, Digit features, pp, cp	83.10
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$, $\lvert$pre$\rvert \leq 3$, Digit features, cp, np	83.70
pw2, pw, cw, nw, nw2, FirstWord, pt, $\lvert$suf$\rvert \leq 3$,$\lvert$pre$\rvert \leq 3$, Digit features, pp, cp, np, nominalPOS, nominalPREP, Gazetteer lists	**89.30**

Table 2: Results on Development Set

identify the best suited set of features for the NER task. From our empirical analysis, we found that the following combination gives the best result with 744 iterations:

F=[$w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$, $\lvert$prefix$\rvert \leq 3$, $\lvert$sufix$\rvert \leq 3$, NE information of the previous word, POS information of the window three, nominalPOS of the current word, nominalPREP, FirstWord, Digit features, Gazetteer lists].

The meanings of the notations, used in experimental results, are defined as below:
cw, pw, nw: Current, previous and next word; pwi, nwi: Previous and the next ith word from the current word; pre, suf: Prefix and suffix of the current word; pt: NE tag of the previous word; cp, pp, np: POS tag of the current, previous and the next word; ppi, npi: POS tag of the previous and the next ith word.

Evaluation results of the system for the development set in terms of overall F-Score (FS) are presented in Table 2. It is observed from Table 2 that word window $[-2, +2]$ gives the best result with 'FirstWord' feature only and the further increase of the window size reduces the overall F-Score value.

Results of Table 2 (3rd and 5th rows) show that the inclusion of NE information of the previous word increases the overall F-Score by 2.68%. It is also indicative from the evaluation results that the performance of the system can be improved by including the prefix and suffix features. Results (6th and 7th rows) also show the fact that prefix and suffix of length upto three of the current word is more effective. In another experiment, it has been also observed that the surrounding word suffixes and/or prefixes do not increase the F-Score value. The overall F-Score value is further improved by 1.8% (7th and 8th rows) with the inclusion of various digit features.

Results (8th and 9th rows) show that POS information of the words improves the overall F-score by 2.1%. In the above experiment, the POS tagger was developed with 26 POS tags. Experimental results (9th, 10th, 11th and 12th rows) suggest that the POS tags of the previous, current and the next words, i.e., POS information of the window $[-1, +1]$ is more effective than POS information of the window $[-2, +2]$, $[-1, 0]$ or $[0, +1]$. In another experiment, we also observed that the POS information of the current word alone is less effective than the window $[-1, +1]$. The modified POS tagger that is developed with 7 POS tags increases the overall F-Score to 85.2%, while other set of features are kept unchanged. So, it can be decided that smaller POS tagset is more effective than the larger POS tagset in NER. We have observed from two separate experiments that the overall F-Score values can further be improved by 0.4% and 0.2%, respectively, with the 'nominalPOS' and 'nominalPREP' features. Finally, an overall F-Score value of 89.3% is obtained by including the gazetteer lists.

The best set of features is identified by training the system with 130K wordforms and testing with the development set of 20K wordforms. Now, the development set is included as part of the training set and resultant training set is thus consists of 150K wordforms. The training set has 20,455 person names, 11,668 location names, 963 organization

names and 11,554 miscellaneous names. We have performed 10-fold cross validation test on this training set. The Recall, Precision and F-Score values for the 10 different experiments in the 10-fold cross validation test are presented in Table 3. The overall average Recall, Precision and F-Score values are 93.8%, 87.8% and 90.7%, respectively.

The other existing Bengali NER systems along with the *baseline* model are also trained and tested under the same experimental setup. The *baseline* model has demonstrated the overall F-Score value of 56.3%. The overall F-Score value of the CRF based NER system is 90.7%, which is an improvement of more than 6% over the HMM based system, best reported Bengali NER system (Ekbal et al., 2007c). The reason behind the rise in overall F-Score value might be its better capability than HMM to capture the morphologically rich and overlapping features of Bengali language. The system has been evaluated also for the four individual NE classes and it has shown the average F-Score values of 91.2%, 89.7%, 87.1% and 99.2%, respectively, for person, location, organization and miscellaneous names.

Test set no.	Recall	Precision	FS (%)
1	92.4	87.3	89.78
2	92.3	87.4	89.78
3	91.4	86.6	88.94
4	95.2	87.7	91.29
5	91.6	86.7	89.08
6	92.2	87.1	89.58
7	94.5	87.9	91.08
8	93.8	89.3	91.49
9	96.9	88.4	92.45
10	97.7	89.6	93.47
Average	93.8	87.8	90.7

Table 3: Results for the 10-fold Cross Validation Test

5 Conclusion

In this paper, we have developed a NER system using CRF with the help of a partially NE tagged Bengali news corpus, developed from the archive of a leading Bengali newspaper available in the web. Experimental results with the 10-fold cross validation test have shown reasonably good Recall, Precision and F-Score values. It has been shown that the contextual window [-2, +2], prefix and suffix of length upto three, first word of the sentence, POS information of the window [-1, +1], current word, NE information of the previous word, different digit features and the various gazetteer lists are the best-suited features for the Bengali NER.

Analyzing the performance using other methods like MaxEnt and Support Vector Machines (SVMs) will be other interesting experiments.

References

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning*, 34(1-3):211–231.

A. Borthwick. 1999. *Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

A. Ekbal and S. Bandyopadhyay. 2007a. Lexical Pattern Learning from Corpus Data for Named Entity Recognition. In *Proceedings of ICON*, pages 123–128, India.

A. Ekbal and S. Bandyopadhyay. 2007b. Pattern Based Bootstrapping Method for Named Entity Recognition. In *Proceedings of ICAPR*, pages 349–355, India.

A. Ekbal and S. Bandyopadhyay. 2007d. A Web-based Bengali News Corpus for Named Entity Recognition. *Language Resources and Evaluation Journal (accepted)*.

A. Ekbal, S.K. Naskar, and S. Bandyopadhyay. 2007c. Named Entity Recognition and Transliteration in Bengali. *Named Entities: Recognition, Classification and Use, Special Issue of Lingvisticae Investigationes Journal*, 30(1):95–114.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289.

Wei Li and Andrew McCallum. 2004. Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. *ACM TALIP*, 2(3):290–294.

A. McCallum and W. Li. 2003. Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of CoNLL*, pages 188–191, Canada.

Fei Sha and Fernando Pereira. 2003. Shallow Parsing with Conditional Random Fields. In *Proceedings of NAACL '03*, pages 134–141, Canada.

An Online Cascaded Approach to Biomedical Named Entity Recognition *

Shing-Kit Chan, Wai Lam, Xiaofeng Yu
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, Hong Kong
{skchan, wlam, xfyu}@se.cuhk.edu.hk

Abstract

We present an online cascaded approach to biomedical named entity recognition. This approach uses an online training method to substantially reduce the training time required and a cascaded framework to relax the memory requirement. We conduct detailed experiments on the BioNLP dataset from the JNLPBA shared task and compare the results with other systems and published works. Our experimental results show that our approach achieves comparable performance with great reductions in time and space requirements.

1 Introduction

In the biomedical domain, the vast amount of data and the great variety of induced features are two major bottlenecks for further natural language processing on the biomedical literature. In this paper, we investigate the biomedical named entity recognition (NER) problem. This problem is particularly important because it is a necessary pre-processing step in many applications.

This paper addresses two main issues that arise from biomedical NER.

Long Training Time: Traditional approaches that depend on the maximum likelihood training method are slow even with large-scale optimization methods such as L-BFGS. This problem worsens with the sheer volume and growth rate of the biomedical literature. In this paper, we propose the use of an online training method that greatly reduces training time.

Large Memory Space: The total number of features used to extract named entities from documents is very large. To extract biomedical named entities, we often need to use extra features in addition to those used in general-purpose domains, such as prefix, suffix, punctuation, and more orthographic features. We need a correspondingly large memory space for processing, exacerbating the first issue. We propose to alleviate this problem by employing a cascaded approach that divides the NER task into a segmentation task and a classification task.

The overall approach is the online cascaded approach, which is described in the remaining sections of this paper: Section 2 describes the general model that is used to address the above issues. We address the issue of long training time in Section 3. The issue of large memory space is addressed in Section 4. Experimental results and analysis are presented in Section 5. We discuss related work in Section 6 and conclude with Section 7.

2 Model Descriptions

Our proposed model is similar to a conditional random field in a sequence labeling task, but we avoid directly dealing with the probability distribution. We use a joint feature representation $\mathbf{F}(\mathbf{x}, \mathbf{y})$ for each

*The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CUHK 4179/03E and CUHK4193/04E) and the Direct Grant of the Faculty of Engineering, CUHK (Project Codes: 2050363 and 2050391). This work is also affiliated with the Microsoft-CUHK Joint Laboratory for Human-centric Computing and Interface Technologies.

input sequence $\mathbf{x}$ and an arbitrary output sequence $\mathbf{y}$, as follows.

$$\mathbf{F}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} \mathbf{f}(\mathbf{x}, \mathbf{y}, i) \quad (1)$$

where each $\mathbf{f}(\mathbf{x}, \mathbf{y}, i)$ is a *local feature function* at position i. For example, in a segmentation task using the IOB2 notation, the k-th *local feature* in $\mathbf{f}(\mathbf{x}, \mathbf{y}, i)$ can be defined as

$$f_k(\mathbf{x}, \mathbf{y}, i) = \begin{cases} 1 & \text{if } x_i \text{ is the word “boy”,} \\ & \text{and } y_i \text{ is the label “B”} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

With parameter $\mathbf{w}$, the best output sequence $\hat{\mathbf{y}}$ for an input sequence $\mathbf{x}$ can be found by calculating the best score:

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}'} \mathbf{w} \cdot \mathbf{F}(\mathbf{x}, \mathbf{y}') \quad (3)$$

3 Online Training

We propose to estimate the parameter $\mathbf{w}$ in an online manner. In particular, we use the online passive-aggressive algorithm (Crammer et al., 2006). Parameters are estimated by margin-based training, which chooses the set of parameters that attempts to make the “margin” on each training instance $(\mathbf{x_t}, \mathbf{y_t})$ greater than a predefined value γ,

$$\mathbf{w} \cdot \mathbf{F}(\mathbf{x_t}, \mathbf{y_t}) - \mathbf{w} \cdot \mathbf{F}(\mathbf{x_t}, \mathbf{y}') \geq \gamma \quad \forall \mathbf{y}' \neq \mathbf{y_t} \quad (4)$$

A *hinge loss* function $\ell(\mathbf{w}; \mathbf{x_t})$ is defined as

$$\ell(\mathbf{w}; \mathbf{x_t}) = \begin{cases} 0 & \text{if } \gamma_t \geq \gamma \\ \gamma - \gamma_t & \text{otherwise} \end{cases} \quad (5)$$

where γ_t is the margin on input $\mathbf{x_t}$ defined as

$$\gamma_t = \mathbf{w} \cdot \mathbf{F}(\mathbf{x_t}, \mathbf{y_t}) - \max_{\mathbf{y}' \neq \mathbf{y_t}} \mathbf{w} \cdot \mathbf{F}(\mathbf{x_t}, \mathbf{y}') \quad (6)$$

In online training, the parameter $\mathbf{w}$ is updated iteratively. Formally speaking, in the t-th iteration with the parameter $\mathbf{w_t}$ and the training instance $\mathbf{x_t}$, we try to solve the following optimization problem.

$$\mathbf{w_{t+1}} = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w_t}\|^{\mathbf{2}} + C\xi \quad (7)$$

such that $\ell(\mathbf{w}; (\mathbf{x_t}, \mathbf{y_t})) \leq \xi$

where $C > 0$ is a user-defined *aggressiveness parameter* and $\xi \geq 0$ is a slack term for the training data when it is not *linearly-separable*. C controls the penalty of the slack term and the *aggressiveness* of each update step. A larger C implies a more aggressive update and hence a higher tendency to overfit. The solution to Problem (7) is

$$\mathbf{w_{t+1}} = \mathbf{w_t} - \tau_t[\mathbf{F}(\mathbf{x_t}, \mathbf{y_t}) - \mathbf{F}(\mathbf{x_t}, \hat{\mathbf{y}}_\mathbf{t})] \quad (8)$$

$$\text{where} \quad \tau_t = \min \left\{ C, \frac{\ell(\mathbf{w_t}; (\mathbf{x_t}, \mathbf{y_t}))}{\|\mathbf{F}(\mathbf{x_t}, \mathbf{y_t}) - \mathbf{F}(\mathbf{x_t}, \hat{\mathbf{y}}_\mathbf{t})\|^2} \right\} \quad (9)$$

The *passiveness* of this algorithm comes from the fact that the parameter $\mathbf{w_t}$ is not updated when the hinge loss for $\mathbf{x_t}$ is zero. It can be proved that the relative loss bound on the training data (and which also bounds the number of prediction mistakes on the training data) cannot be much worse than the best fixed parameter chosen in hindsight. See (Crammer et al., 2006) for a detailed proof.

Following most of the work on margin-based training, in this paper we choose γ to be a function of the correct output sequence $\mathbf{y}$ and the predicted output sequence $\hat{\mathbf{y}}$.

$$\gamma(\mathbf{y}, \hat{\mathbf{y}}) = \begin{cases} 0 & \text{if } \mathbf{y} = \hat{\mathbf{y}} \\ \sum_{i=1}^{|\mathbf{y}|} [[y_i \neq \hat{y}_i]] & \text{otherwise} \end{cases} \quad (10)$$

where $[[z]]$ is 1 if z is true, and 0 otherwise.

The major computation difficulty in this online training comes from Equation (3). Finding the best output $\hat{\mathbf{y}}$ is in general an intractable task. We follow the usual first-order independence assumption made in a linear-chained CRF (Lafferty et al., 2001) model and calculate the best score using the Viterbi algorithm.

4 Cascaded Framework

We divide the NER task into a segmentation task and a classification task. In the segmentation task, a sentence $\mathbf{x}$ is segmented, and possible segments of biomedical named entities are identified. In the classification task, the identified segments are classified into one of the possible named entity types or rejected.

In other words, in the segmentation task, the sentence $\mathbf{x}$ are segmented by

$$\hat{\mathbf{y}}_{\mathbf{s}} = \operatorname*{argmax}_{\mathbf{y}'} \mathbf{w}_{\mathbf{s}} \cdot \mathbf{F}_{\mathbf{s}}(\mathbf{x}, \mathbf{y}') \quad (11)$$

where $\mathbf{F}_{\mathbf{s}}(\cdot)$ is the set of segment features, and $\mathbf{w}_{\mathbf{s}}$ is the parameter for segmentation.

In the classification task, the segments (which can be identified by $\mathbf{y}_{\mathbf{s}}$) in a sentence $\mathbf{x}$ are classified by

$$\hat{\mathbf{y}}_{\mathbf{c}} = \operatorname*{argmax}_{\mathbf{y}'} \mathbf{w}_{\mathbf{c}} \cdot \mathbf{F}_{\mathbf{c}}(\mathbf{x}, \mathbf{y}_{\mathbf{s}}, \mathbf{y}') \quad (12)$$

where $\mathbf{F}_{\mathbf{c}}(\cdot)$ is the set of classification features, and $\mathbf{w}_{\mathbf{c}}$ is the parameter for classification.

In this cascaded framework, the number of possible labels in the segmentation task is N_s. For example, $N_s = 3$ in the IOB2 notation. In the classification task, the number of possible labels is $N_c + 1$, which is the number of entity types and one label for "Other". Following the first-order independence assumption, the maximum total number of features in the two tasks is $O(\max(N_s^2, N_c^2))$, which is much smaller than the single-phase approach in which the total number of features is $O((N_s N_c)^2)$.

Another potential advantage of dividing the NER task into two tasks is that it allows greater flexibility in choosing an appropriate set of features for each task. In fact, adding more features may not necessarily increase performance. (Settles, 2004) reported that a system using a subset of features outperformed one using a full set of features.

5 Experiments

We conducted our experiments on the GENIA corpus (Kim et al., 2003) provided in the JNLPBA (Kim et al., 2004) shared task[1]. There are 2,000 MEDLINE abstracts in the GENIA corpus with named entities tagged in the IOB2 format. There are 18,546 sentences and 492,551 words in the training set, and 3,856 sentences and 101,039 words in the evaluation set. The line indicating the MEDLINE abstract ID boundary information is not used in our experiments. Each word is tagged with "B-X", "I-X", or "O" to indicate that the word is at the "beginning" (B) or "inside" (I) of a named entity of type X, or "outside" (O) of a named entity. The named entity types are: DNA, RNA, cell_line, cell_type, and protein.

System	F_1
(Zhou and Su, 2004)	72.55
Online Cascaded	**72.16**
(Okanohara et al., 2006)	71.48
(Kim et al., 2005)	71.19
(Finkel et al., 2004)	70.06
(Settles, 2004)	69.80

Table 1: Comparisons with other systems on overall performance (in percentage).

5.1 Features

The features used in our experiments mainly follow the work of (Settles, 2004) and (Collins, 2001). For completeness, we briefly describe the features here. They include word features, orthographic features, parts-of-speech (POS), and two lexicons. The word features include unigram, bigram, and trigram (e.g. the previous word, the next word, and the previous two words), whereas the orthographic features include capital letter, dash, punctuation, and word length. *Word class* (WC) features are also added, which replace a capital letter with "A", a lower case letter with "a", a digit with "0", and all other characters with "_". Similar *brief word class* (BWC) features are added by collapsing all of the consecutive identical characters in the *word class* features into one character. For example, for the word `NF-kappa`, WC = `AA_aaaaa`, and BWC = `A_a`. These are listed in Tables 2 and 3. The POS features are added by the GENIA tagger[2].

All of these features except for the prefix/suffix features are applied to the neighborhood window $[i-1, i+1]$ for every word. Two lexicons for cell lines and genes are drawn from two online public databases: the Cell Line Database[3] and the BBID[4]. The prefix/suffix and lexicon features are applied to position i only. All of the above features are com-

[1] http://research.nii.ac.jp/~collier/workshops/JNLPBA04st.htm

[2] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/

[3] http://www.biotech.ist.unige.it/cldb/cname-tz.html

[4] http://bbid.grc.nia.nih.gov/bbidgene.html

Unigram	(w_{-2}), (w_{-1}), (w_0), (w_1), (w_2)
Bigram	$(w_{-2}\ w_{-1})$, $(w_{-1}\ w_0)$, $(w_0\ w_1)$, $(w_1\ w_2)$
Trigram	$(w_{-2}\ w_{-1}\ w_0)$, $(w_{-1}\ w_0\ w_1)$, $(w_0\ w_1\ w_2)$

Table 2: Word features used in the experiment: w_0 is the current word, w_{-1} is the previous word, etc.

Word features	as in Table 2
Prefix/suffix	Up to a length of 5
Word Class	WC
Brief Word Class	BWC
Capital Letter	`^[A-Z][a-z]` `[A-Z]{2,}` `[a-z]+[A-Z]+`
Digit	`[0-9]+` `^[^0-9]*[0-9][^0-9]*$` `^[^0-9]*[0-9][0-9][^0-9]*$` `^[0-9]+$` `[0-9]+[,.][0-9,.]+` `[A-Za-z]+[0-9]+` `[0-9]+[A-Za-z]+`
Dash	`[-]+` `^[-]+` `[-]+$`
Punctuation	`[,;:?!-+'"\/]+`
Word length	length of the current word x_i

Table 3: Features used in the JNLPBA experiment. The features for *Capital Letter*, *Digit*, *Dash*, and *Punctuation* are represented as regular expressions.

bined with the previous label y_{i-1} and the current label y_i to form the final set of features.

In the segmentation task, only three labels (i.e. `B, I, O`) are needed to represent the segmentation results. In the classification task, the possible labels are the five entity types and "`Other`". We also add the segmentation results as features in the classification task.

5.2 Results

We tried different methods to extract the named entities from the JNLPBA dataset for comparisons. These programs were developed based on the same basic framework. All of the experiments were run on a Unix machine with a 2.8 GHz CPU and 16 GB RAM. In particular, the CRF trained by maximum-likelihood uses the L-BFGS algorithm (Liu and Nocedal, 1989), which converges quickly and gives a good performance on maximum entropy models (Malouf, 2002; Sha and Pereira, 2003). We compare our experimental results in several dimensions.

Training Time: Referring to Table 4, the training time of the online cascaded approach is substantially shorter than that of all of the other approaches. In the single-phase approach, training a CRF by maximum likelihood (ML) using the L-BFGS algorithm is the slowest and requires around 28 hours. The online method greatly reduces the training time to around two hours, which is 14 times faster. By employing a two-phase approach, the training time is further reduced to half an hour.

Memory Requirement: Table 4 shows the number of features that are required by the different methods. For methods that use the single-phase approach, because the full set of features (See Section 4) is too big for practical experiments on our machine, we need to set a higher cutoff value to reduce the number of features. With a cutoff of 20 (i.e. only features that occur more than 20 times are used), the number of features can still go up to about 8 million. However, in the two-phase approach, even with a smaller cutoff of 5, the number of features can still remain at about 8 million.

F_1**-measure:** Table 4 shows the F_1-measure in our experiments, and Table 1 compares our results with different systems in the JNLPBA shared tasks and other published works[5]. Our performance of the single-phase CRF with maximum likelihood training is 69.44%, which agrees with (Settles, 2004) who also uses similar settings. The single-phase online method increases the performance to 71.17%. By employing a cascaded framework, the performance is further increased to 72.16%, which can be regarded as comparable with the best system in the JNLPBA shared task.

6 Related Work

The online training approach used in this paper is based on the concept of "margin" (Cristianini, 2001). A pioneer work in online training is the perceptron-like algorithm used in training a hidden Markov model (HMM) (Collins, 2002). (McDonald

[5]We are aware of the high F_1 in (Vishwanathan et al., 2006). We contacted the author and found that their published result may be incomplete.

Experiments		no. of features	training time	F_1	rel. err. red. on F_1
single-phase	CRF + ML	8,004,392	1699 mins	69.44	–
	CRF + Online	8,004,392	116 mins	71.17	5.66%
two-phase	Online + Cascaded	seg: 2,356,590 class: 8,278,794	14 + 15 = 29 mins	72.16	8.90%

Table 4: The number of features, training time, and F_1 that are used in our experiments. The cutoff thresholds for the single-phase CRFs are set to 20, whereas that of the online cascaded approach is set to 5 in both segmentation and classification. The last column shows the relative error reductions on F_1 (compared to CRF+ML).

Experiments	R	P	F_1
Segmentation	80.13	73.68	76.77
Classification	92.75	92.76	92.76

Table 5: Results of the individual task in the online cascaded approach. The F_1 of the classification task is 92.76% (which is based on the fully correct segmented testing data).

et al., 2005) also proposed an online margin-based training method for parsing. This type of training method is fast and has the advantage that it does not need to form the dual problem as in SVMs. A detailed description of the online passive-aggressive algorithm used in this paper and its variants can be found in (Crammer et al., 2006). The Margin Infused Relaxed Algorithm (MIRA), which is the ancestor of the online passive-aggressive algorithm and mainly for the *linearly-separable* case, can be found in (Crammer and Singer, 2003).

(Kim et al., 2005) uses a similar two-phase approach but they need to use rule-based post-processing to correct the final results. Their CRFs are trained on a different dataset that contains all of the other named entities such as *lipid*, *multi cell*, and *other organic compound*. Table 1 shows the comparisons of the final results.

In the JNLPBA shared task, eight NER systems were used to extract five types of biomedical named entities. The best system (Zhou and Su, 2004) uses "deep knowledge", such as name alias resolution, cascaded entity name resolution, abbreviation resolution, and in-domain POS. Our approach is relatively simpler and uses a unified model to accomplish the cascaded tasks. It also allows other post-processing tasks to enhance performance.

7 Conclusion

We have presented an online cascaded approach to biomedical named entity recognition. This approach substantially reduces the training time required and relaxes the memory requirement. The experimental results show that our approach achieves performance comparable to the state-of-the-art system.

References

Michael Collins. 2001. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 489–496.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Nello Cristianini. 2001. Support vector and kernel machines. ICML tutorial. Available at http://www.support-vector.net/icml-tutorial.pdf.

J. Finkel, S. Dingare, H. Nguyen, M. Nissim, C. Manning, and G. Sinclair. 2004. Exploiting context for biomedical entity recognition: from syntax to the web. In *Proceedings of the International Joint Workshop on*

Natural Language Processing in Biomedicine and its Applications (NLPBA), pages 88–91.

J.d. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. Genia corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics (Supplement: Eleventh International Conference on Intelligent Systems for Molecular Biology)*, 19:180–182.

J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In N. Collier, P. Ruch, and A. Nazarenko, editors, *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland*, pages 70–75, August 28–29. held in conjunction with COLING'2004.

Seonho Kim, Juntae Yoon, Kyung-Mi Park, and Hae-Chang Rim. 2005. Two-phase biomedical named entity recognition using a hybrid method. In *Proceedings of The Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, pages 646–657.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289.

D. C. Liu and J. Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL-2002*, pages 49–55.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98.

Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 465–472.

B. Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 104–107.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 969–976.

GuoDong Zhou and Jian Su. 2004. Exploring deep knowledge resources in biomedical name recognition. In *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 99–102.

Automatic rule acquisition for Chinese intra-chunk relations

Qiang Zhou
Center for Speech and Language Technologies, Division of Technical Innovation and Development
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University, Beijing 100084, P. R. China
zq-lxd@mail.tsinghua.edu.cn

Abstract

Multiword chunking is defined as a task to automatically analyze the external function and internal structure of the multiword chunk(MWC) in a sentence. To deal with this problem, we proposed a rule acquisition algorithm to automatically learn a chunk rule base, under the support of a large scale annotated corpus and a lexical knowledge base. We also proposed an expectation precision index to objectively evaluate the descriptive capabilities of the refined rule base. Some experimental results indicate that the algorithm can acquire about 9% useful expanded rules to cover 86% annotated positive examples, and improve the expectation precision from 51% to 83%. These rules can be used to build an efficient rule-based Chinese MWC parser.

1 Introduction

In recent years, the chunking problem has become a hot topic in the communities of natural language processing. From 2000 to 2005, several different chunking-related tasks, such as text chunking (Sang and Buchholz, 2000), clause identification (Sang and Dejean, 2001), semantic role labeling (Carreras and Marquez, 2005), were defined in the CoNLL conferences. Much research has been devoted to the problem through different points of view.

Many computational linguists regard chunking as a shallow parsing technique. Due to its efficiency and robustness on non-restricted texts, it has become an interesting alternative to full parsing in many NLP applications. On the base of the chunk scheme proposed by Abney (1991) and the BIO tagging system proposed in Ramshaw and Marcus(1995), many machine learning techniques are used to deal with the problem. However, almost all the chunking systems focus on the recognition of non-overlapping cores of chunks till now, none of them care about the internal structure analysis of chunks.

In our opinion, the internal structure of a chunk, including its head and the dependency relation between head and other components, plays an important role for semantic content understanding for the chunk. They are especially useful for the languages with few morphological inflections, such as the Chinese language. Therefore, we design a multiword chunking task to recognize different multiword chunks (MWCs) with the detailed descriptions of external function and internal structure in real texts. Its main difficulty lies in the preciously identification of different lexical relationships among the MWC components. Some detailed lexical semantic knowledge is required in the task.

To deal with this problem, we proposed a rule acquisition algorithm to automatically learn a MWC rule base, under the support of a large scale annotated corpus and a lexical knowledge base. We also proposed an expectation precision index to evaluate the descriptive capabilities of the refined rule base. Some experimental results indicate that our current algorithm can acquire about 9% useful expanded rules to cover 86% annotated positive examples, and improve the expectation precision from 51% to 83%.

2 Multiword chunking task

Informally, a MWC is a chunk with two or more words, where each word links to a semantic head through different dependency relations. Four syntactic dependency relationships are used in the paper: (1) Modifier-Head relation, (2) Predicate-

Object relation,(3) Predicate-Compliment relation, (4) Coordinate relation. They can determinate the following functional position tags for each word in a MWC: (1) M--Modifier; (2) H--Head; (3) P--Predicate; (4) O--Object; (5) C--Compliment; (6) J--Coordinate constituent. Based on them, we define three topological constructions as follows:

(1) Left-Corner-Centre (LCC) construction

All the words in a chunk link to the left-corner head and form a left-head dependency structure. Its basic pattern is: H C_1 … C_n. The typical dependencies among them are Predicate-Object or Predicate-Compliment relations: $C_1 \rightarrow H$, … , $C_n \rightarrow H$. They form the following functional tag serial : P [C|O].

(2) Right-Corner-Centre (RCC) construction

All the words in a chunk link to the right-corner head and form a right-head dependency structure. Its basic pattern is: A_1 … A_n H. The typical dependencies among them are Modifier-Head relations: $A_1 \rightarrow H$, … , $A_n \rightarrow H$. They form the following functional tag serial : $\{M\}_+$ H.

(3) Chain Hooking (CH) construction

Each word in a chunk links to its right-adjacent word. All of them form a multi-head hooking chain. Its basic pattern is: H_0 H_1 … H_n, where H_i, $i \in [1,n-1]$ is the chain head in differnt levels, H_n is the semantic head of the overall chunk. The typical dependencies among them are Modifier-Head or Coordinate relations : $H_0 \rightarrow H_1$, … , $H_{n-1} \rightarrow H_n$. They form the following functional tag serial : $\{J\}_*$ or [M|J] $\{K|J\}_*$ H, where K represents the internal chain head.

We think the above three constructions can cover almost all important syntactic relations in real text sentences. Now, we can give a formal definition for a multiword chunk.

Definition: two or more words can form a multiword chunk if and only if it has one of the above three internal topological constructions.

The MWC definition builds the one-to-one corresponding between the word serials with different function tags and their dependency structure. So we can easily describe some MWCs with complex nested structures. In the paper, we add a further restriction that each MWC can only comprise the content words, such as nouns, verbs, adjectives, etc. This restriction can make us focus on the analysis of the basic content description units in a sentence.

Each MWC is assigned two tags to describe its external function and internal structure. For example, a 'np-ZX' MWC represents a noun chunk with internal modifier-head relationship. Table 1 lists all the function and relation tags used in our MWC system. The np, mp, tp, sp form as the nominal chunk set. Their typical relation tags are ZX, LN and LH. The vp and ap form as the predicate chunk set. Their typical relation tags are ZX, PO, SB and LH.

F-tags	Descriptions	R-tags	Descriptions
np	noun chunk	ZX	modifier-head relationship
vp	verb chunk	PO	verb-object relationship
ap	adjective chunk	SB	verb-compliment relationship
mp	quantity chunk	LH	Coordinate relationship
sp	space chunk	LN	chain hooking relationship
tp	time chunk		

Table 1 Function and relation tags of MWCs

The following is a MWC annotated sentence:

[tp-ZX 长期/t(long time) 以来/f(since)] ，/w 他/r(he) 为/p(for) 维护/v(safeguard) [np-ZX 世界/n (world) 和平/n(peace)] 的/u [np-ZX 崇高/a(lofty) 事业/n(undertaking)] [vp-PO 倾注/v (devote) 心血/n (painstaking)] ，/w 作出/v(make) 了/u 卓越/a(outstanding) 的/u 贡献/v (contribution) 。/w[1] (For a long time past, he has devoted all his energy into the lofty undertaking to safeguard world peace and made a outstanding contribution.) (1)

There are four MWCs in the sentence. From which, we can easily extract the positive and negative examples for a MWC rule. For example, in the sentence, we can extract a positive example: 倾注/v (devote) 心血/n (painstaking), and a negative example: 维护/v(safeguard) 世界/n (world) for the verb chunk rule : v+n→ vp-PO.

3 Automatic rule acquisition

The goal of the rule acquisition algorithm is to

[1] POS tags used in the sentence: t-time noun, f-direction, r-pronoun, p-preposition, v-verb, n-noun, u-auxilary, a-adjective, d-adverb, w-puntuation.

automatically acquire some syntactic structure rules to describe which words in which context in a sentence can be reduced to a reliable MWC, on the base of a large scale annotated corpus and a lexical knowledge base.

Each rule will have the following format: <structure description string> → <reduced tag> <confidence score>

Two types of structural rules are used in our algorithm: (1) Basic rules, where only POS tags are used in the components of a structure rule; (2) Expanded rules, where some lexical and contextual constraint is added into the structure rule string to give more detailed descriptions. The reduced tag has two kinds of MWC tags that are same as ones defined in Table 1.

Each rule consists of all the positive and negative examples covered by the rule in the annotated corpus. For the word serial matched with the structure description string of a rule, if it can be reduced as a MWC in the annotated sentence, it can be regarded as a positive example. Otherwise, it is a negative example. All of them form a special state space for each acquired rule. Therefore, the confidence score (θ) for the rule can be easily computed to evaluate the accuracy expectation to apply it in an automatic parser. Its computation formula is: $\theta = f_P / (f_P + f_N)$, where f_P is the frequency of the positive examples, and f_N is the frequency of the negative examples.

A two-step acquisition strategy is adopted in our algorithm.

The first step is rule learning. We firstly extract all basic rules with positive examples from the annotated corpus. Then, we match the extracted structure string of each basic rule in all the corpus sentences to find all possible negative examples and build state space for it. Through rule reliability computation (see the following section), we can extract all high-reliability basic rules as the final result, and all other basic rules with higher frequency for further rule refinement.

The second step is rule refining. We gradually expand each rule with suitable lexical and contextual constraint based on an outside lexical knowledge base, dynamically divide and automatically allocate its positive and negative examples into the expanded rules and form different state spaces for them. From them, we can extract all the high and middle reliability expanded rules as the final results.

At last, by combining all the extracted basic and expanded rules, we build a hierarchical acquired rule base for parser application.

Two key techniques are proposed in the algorithm:

(1) Rule reliability evaluation

The intuition assumption is that: if a rule has a higher confidence score and can cover more positive examples, then it can be regarded as a reliable rule.

Types	Decision conditions
1	● $(f_P>=10)$ && $(\theta>=0.85)$ ● $((f_P>=5)$ && $(f_P<10))$ && $(\theta>=0.9)$ ● $((f_P>=2)$ && $(f_P<5))$ && $(\theta>=0.95)$
2	● $(f_P>=10)$ && $(\theta>=0.5)$ ● $((f_P>=5)$ && $(f_P<10))$ && $(\theta>=0.55)$ ● $((f_P>=2)$ && $(f_P<5))$ && $(\theta>=0.6)$ ● $(f_P>0)$ && $(\theta>=0.6)$
3	● $(f_P>=10)$ && $(\theta>=0.1)$ ● $((f_P>=5)$ && $(f_P<10))$ && $(\theta>=0.2)$ ● $((f_P>=2)$ && $(f_P<5))$ && $(\theta>=0.3)$ ● $(f_P>0)$ && $(\theta>=0.3)$
4	All others

Table 2 Four reliability types of the acquired rules

By setting different thresholds for θ and f_P, we can classify all acquired rules into the following four types of rule sets: (1) high-reliability (HR) rules; (2) middle-reliability (MR) rules; (3) low-reliability rules; (4) Useless and noise rules. Table 2 shows different decision conditions for them in our current algorithm. Based on this uniform evaluation standard, we can easily extract effective rules from different acquired rule base and quickly exclude useless noise rules.

(2) Rule expansion and refinement

When a rule is not reliable enough, the expansion step is set off: new knowledge is added to the rule in order to constrain it. The purpose is to dynamically divide the state space of the rule and reduce the proportion of negative examples covered by the current rule. For every annotated positive or negative example, our expansion strategy is as follows:

Firstly, we expand a rule description through looking up different lexical knowledge base. For the verb chunks with LCC constructions, we use the following lexical constraint: (1) Lexical-syntactic relation pairs, (2) Subcategory frame of

head verb. For the noun chunks with RCC and CH constructions, we use the following lexical constraint: (1) Lexical-syntactic relation pairs, (2) Semantic class of head noun.

Secondly, we expand a rule description example with or without lexical constraint through looking up its left and right adjacent contexts. For each rule waiting for expansion, we add its left-adjacent POS tag, right-adjacent POS tag, left and right adjacent POS tag to form three expanded rule with contextual constraint.

For example, for the positive example "倾注/v (devote) 心血/n (painstaking) " of "v+n" rule in the above sentence (1), we can get the following expanded rules:

- v(WC-L)+n(WC-R) // + v-n relationship pair
- v(winl:VNPLIST)+n // + verb subcate frame
- n__v+n // + left POS constraint
- v+n__w // +right POS constraint
- n__v+n__w // +l and +r POS constraint

They can be put into the state space pool as the expanded rules with positive example information for frequency calculation.

Unlike the *information-gain* measure used in FOIL system (Quinlan, 1990), we do not impose any criteria for selecting different knowledge. All the suitable expanded rules are selected through the final confidence score evaluation indexes.

4 Experimental results

All the news files with about 200,000 words in the Chinese treebank TCT (Zhou, 2004) were selected as the experimental data. They were separated into two data sets: (1) training set, which consists of about 80% data and is used for rule acquisition; (2) test set, which consists of about 20% data and is used for parser evaluation.

Then we automatically extracted all the MWCs from the annotated trees and built two MWC banks. Among them, 76% are noun chunks and verb chunks. They are the key points for rule acquisition and parsing application. In the training set, about 94% verb chunks are two-word chunks. But for noun chunks, the percentage of two-word chunks is only 76%. More than 24% noun chunks comprise three or more words. The complexities of noun chunks bring more difficulties for rule acquisition and automatic MWC parsing.

We also used the following lexical knowledge base for rule expansion and refinement: (1) Lexical relationship base. It consists of 966953 lexical pairs with different syntactic relationships. All the data are extracted from 4 different language resources. (2) Verb subcategory data. It consists of 5712 verbs with the "v+np" subcat frames and 1065 verbs with the "v+vp" subcat frames. All the data are extracted from a Chinese grammatical dictionary (Yu and al., 1998). (3) Noun thesaura data. It consists of 26906 nouns annotated with the different semantic types All the data are extracted from Hownet-2000[2].

4.1 Rule base acquisition

We ran our algorithm on the above language resources and obtained the following results.

In the rule learning stage, we extracted 735 basic rules from the training set. After reliability evaluation, we obtained 61 HR rules and 150 less reliable rules for further refinement. Although these 211 rules only make up 29% of all the 735 acquired rules, they cover about 97% positive examples in the training set. Thus, almost all the useful information can be reserved for further rule expansion and refinement.

In the rule refining stage, 47858 rules were expanded from the 150 basic rules. Among them, all 2036 HR and 2362 MR rules were selected as the final results. They make up about 9% of all the expanded rules, but cover 86% positive examples. It indicates the effectiveness of our current rule acquisition algorithm.

In order to evaluate the descriptive capability of the acquired rules objectively, we proposed an expectation precision (*EP*) index to estimate the parsing accuracy when we apply the acquired rules to all the positive examples in the training set. Its computation formula is as follows:

$$EP = \sum_{i=1}^{N} (f_{Pi} * \theta_i) / \sum_{i=1}^{N} f_{Pi}$$

where N is the total number of the rules in a rule base, f_{Pi} and θ_i are the positive example frequency and confidence score of the i^{th} rule in the rule base. An intuition assumption behind the *EP* definition is that a rule base with higher *EP* index will imply its better descriptive capability for some special linguistic phenomena. Therefore, its better parsing performance in a rule-based parser can be expected. To prove this assumption, we designed a

[2] The data is available in http://www.keenage.com

simple comparison experiment to analyze the improvement effects of different lexical and contextual constraint used in our expanded rules.

We divided all 150 basic rules into 4 subsets, according to their different internal structure characteristics: (1) Noun chunks with RCC and CH constructions; (2) Verb chunks with LCC constructions; (3) Verb chunks with RCC constructions; (4) All other MWCs.

The rules in the subset 1 and 2 cover majority of the positive examples in the training set. They have complex internal structures and lexical relations. So we applied the lexical knowledge base and contextual constraint to expand them. Comparatively, the rules in subset 3 and 4 have simpler structures, so we only used the contextual constraint to expand them.

Table 3 shows the *EP* indexes of these rule subsets before and after rule refining. For all 150 basic rules, after rule expansion and refinement, the *EP* index was improved about 65%. For the simpler structure rules in subset 3 and 4, just the application of contextual constraint can bring dramatic improvement in the *EP* index. It indicates the importance of the local contextual information for multiword chunk recognition.

Sub set	Rule sum	Covered positive examples	EP before expansion (%)	EP after expansion (%)
1	51	13689	52.70	81.40
2	20	8859	45.14	80.56
3	24	2342	28.12	93.27
4	55	3566	66.85	93.22
Total	150	28456	50.56	83.36

Table 3 Descriptive capability analysis of different kinds of expanded rule sets

For the major subset 1 and 2, *EP* index also shows great improvement. It increased about 54% and 78% in the subset 1 and 2 respectively. As we can see, the applying effects of lexical and contextual constraint on the verb chunks were superior to that on the noun chunks. Two factors contribute to this phenomenon. First, the simpler internal structures of most verb chunks guarantee the availability of almost all corresponding lexical relationship pairs. Second, most lexical pairs used in verb chunks have stronger semantic relatedness than that in noun chunks.

4.2 Parsing performance evaluation

Based on the rule base automatically acquired through the above algorithm, we developed a rule-based MWC parser to automatically recognize different kinds of MWCs in the Chinese sentences after word segmentation and POS tagging. Through θ-based disambiguation technique, the parser can output most reliable MWCs in the disambiguated region of a sentence and keep some ambigous regions with less reliable MWC structures to provide multiple selection possibilities for a full syntactic parser. Some detailed information of the parser can be found in (Zhou, 2007).

We used three commonly-used indexes : precision, recall and F-measure to evaluate the performance of the parser. Two different criteria were set to determinate the correctness of a recognized MWC. (1) 'B+F+R' criterion : It must have the same left and right boundaries, function tag and relation tag as that of the gold standard. (2) 'B+F' criterion : It must have the same left and right boundaries, function tag as that of the gold standard.

Table 4 shows the experimental results under the disambigutated regions, which cover 95% of the test data.

Type	'B+F+R' criterion	'B+F' criterion
np	75.25/75.76/75.50	83.68/84.25/83.97
vp	83.23/81.46/82.34	87.35/85.49/86.41
mp	94.89/95.26/95.08	94.89/95.26/95.08
ap	93.99/97.33/95.63	93.99/97.33/95.63
tp	92.75/88.18/90.40	93.52/88.92/91.16
sp	78.76/86.41/82.41	79.65/87.38/83.33
Total	81.76/81.44/81.60	87.01/86.67/86.84

Table 4 Open test results (P/R/F-m, %) under the disambiguated regions

The differences of F-measures among three MWC subsets, i.e. noun chunks, verb chunks and other chunks, show interesting positive association with the differences of their *EP* indexes listed in the previous sections. When we apply the acquired rule base with higher *EP* index in the rule-based parser, we can get better parsing performance. It indicates that *EP* value can be used as an important objective index to evaluate the descriptive capability of the rule base automatically acquired for large scale annotated corpus.

The lower F-measure of noun and verb chunk

under 'B+F+R' criterion shows the difficulty for lexical relation recognition, especially for the complex noun chunks. There are still much improvement room in future research.

5 Related work

In the area of chunking rule acquisition and refinement, several approaches have been proposed. Cardie and Pierce(1999) explored the role of lexicalization and pruning of grammars for base noun phrase identification. Their conclusion is that error-driven pruning is a remarkably robust method for improving the performance of grammar rules. Dejean(2002) proposed a top-down inductive system, ALLis, for learning and refining linguistic structures on the base of contextual and lexicalization knowledge extracted from an annotated corpus. Choi et all(2005) proposed a method for automatically extracting partial parsing rules from a tree-annotated corpus using decision tree induction. The acquired grammar is similar to a phrase structure grammar, with contextual and lexical information, but it allows building structures of depth one or more.

All these researches prove the important role of lexical and contextual information for improving the rule descriptive capability. However, the lexical information used in these systems is still restricted in the lexical head of a constituent. None of the lexical relationship knowledge extracted from the annotated corpus or other outside language resources has been applied. Therefore, the room for improvement of the rule descriptive capability is restricted to a certain extent.

6 Conclusions

Three main contributions of the paper are summarized as follows. (1) We design a new multiword chunking task. Based on the topological structure definition, we establish the built-in relations between multiword chunk examples in annotated corpus and lexical relationship pairs in outside lexical knowledge base. (2) We propose an efficient algorithm to automatically acquire hierarchical structure rules from large-scale annotated corpus. By introducing different kinds of lexical knowledge coming from several different language resources, we set up an open learning environment for rule expansion and refinement. (3) We propose an expectation precision index to evaluate the descriptive capability of the refined rule base. Experimental results show that it has stronger positive association with the F-measure of parser performance evaluation.

Acknowledgements. The research was supported by NSFC (Grant No. 60573185, 60520130299). Thank the comments and advice of the anonymous reviewers.

References

Steven Abney. 1991. Parsing by Chunks. In *R. Berwick, S. Abney and C. Tenny (eds.) Principle-Based Parsing, Kluwer Academic Publishers.*

Claire Cardie and D. Pierce. 1999. The Role of Lexicalization and Pruning for Base Noun Phrase Grammars. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (AAAI-99).

X. Carreras and L. M`arquez. 2005. Introduction to the conll-2004 shared tasks: Semantic role labeling. In Proc. of CoNLL-2005.

Myung-Seok Choi, Chul Su Lim, and Key-Sun Choi. 2005. Automatic Partial Parsing Rule Acquisition Using Decision Tree Induction. In *R. Dale et al. (Eds.). Proc. of IJCNLP 2005, Seoul, Korea* . p143–154.

Herve Dejean. 2002 Learning rules and their exceptions. *Journal of Machine Learning Research*, 2002: 669–693.

J R. Quinlan 1990. Learning logical definitions from relations. *Machine Learning*, 5:239–266.

L Ramshaw and M Marcus. 1995.Text chunking using transformation-based learning. In *Proc. of the Third Workshop on Very Large Corpora*, p82-94.

Erik F. Tjong Kim Sang and S. Buchholz 2000 Introduction to CoNLL-200 Shared Task: Chunking. In *Proc. of CoNLL-2000 and LLL-2000*. Lisbon. p127-132.

Erik F. Tjong Kim Sang and H. Déjean 2001. Introduction to the CoNLL-2001 Shared Task: Clause Identification. In *Proc. of CoNLL-2001*, Toulouse, France, p53-57.

Shiwen Yu, Xuefeng Zhu, et al. 1998 *A Complete Specification of the Grammatical Knowledge-base of Contemporary Chinese*. Tsinghua University Press. (in Chinese)

Qiang Zhou 2004. Annotation scheme for Chinese Treebank. *Journal of Chinese Information Processing*, 18(4): 1-8. (in Chinese)

Qiang Zhou. 2007. A rule-based Chinese chunk parser. In *Proc. Of ICCC-2007,* furthercoming.

Japanese Named Entity Recognition Using Structural Natural Language Processing

Ryohei Sasano*
Graduate School of Information Science and Technology, University of Tokyo
ryohei@nlp.kuee.kyoto-u.ac.jp

Sadao Kurohashi
Graduate School of Infomatics, Kyoto University
kuro@i.kyoto-u.ac.jp

Abstract

This paper presents an approach that uses structural information for Japanese named entity recognition (NER). Our NER system is based on Support Vector Machine (SVM), and utilizes four types of structural information: cache features, coreference relations, syntactic features and caseframe features, which are obtained from structural analyses. We evaluated our approach on CRL NE data and obtained a higher F-measure than existing approaches that do not use structural information. We also conducted experiments on IREX NE data and an NE-annotated web corpus and confirmed that structural information improves the performance of NER.

1 Introduction

Named entity recognition (NER) is the task of identifying and classifying phrases into certain classes of named entities (NEs), such as names of persons, organizations and locations.

Japanese texts, which we focus on, are written without using blank spaces. Therefore, Japanese NER has tight relation with morphological analysis, and thus it is often performed immediately after morphological analysis (Masayuki and Matsumoto, 2003; Yamada, 2007). However, such approaches rely only on local context. The Japanese NER system proposed in (Nakano and Hirai, 2004), which achieved the highest F-measure among conventional systems, introduced the *bunsetsu*[1] feature in order to consider wider context, but considers only adjacent *bunsetsus*.

On the other hand, as for English or Chinese, various NER systems have explored global information and reported their effectiveness. In (Malouf, 2002; Chieu and Ng, 2002), information about features assigned to other instances of the same token is utilized. (Ji and Grishman, 2005) uses the information obtained from coreference analysis for NER. (Mohit and Hwa, 2005) uses syntactic features in building a semi-supervised NE tagger.

In this paper, we present a Japanese NER system that uses global information obtained from several structural analyses. To be more specific, our system is based on SVM, recognizes NEs after syntactic, case and coreference analyses and uses information obtained from these analyses and the NER results for the previous context, integrally. At this point, it is true that NER results are useful for syntactic, case and coreference analyses, and thus these analyses and NER should be performed in a complementary way. However, since we focus on NER, we recognize NE after these structural analyses.

2 Japanese NER Task

A common standard definition for Japanese NER task is provided by IREX workshop (IREX Committee, 1999). IREX defined eight NE classes as shown in Table 1. Compared with the MUC-6 NE task definition (MUC, 1995), the NE class "ARTIFACT," which contains book titles, laws, brand names and so on, is added.

NER task can be defined as a chunking problem to identify token sequences that compose NEs. The chunking problem is solved by annotating chunk tags to tokens. Five chunk tag sets, IOB1, IOB2, IOE1, IOE2 and IOBES are commonly used. In this paper, we use the IOBES model, in which "S" denotes a chunk itself, and "B," "I" and "E" denote the

*Research Fellow of the Japan Society for the Promotion of Science (JSPS)

[1] *Bunsetsu* is a commonly used linguistic unit in Japanese, consisting of one or more adjacent content words and zero or more following functional words.

Table 1: Definition of NE in IREX.

NE class	Examples
ORGANIZATION	NHK Symphony Orchestra
PERSON	Kawasaki Kenjiro
LOCATION	Rome, Sinuiju
ARTIFACT	Nobel Prize
DATE	July 17, April this year
TIME	twelve o'clock noon
MONEY	sixty thousand dollars
PERCENT	20%, thirty percents

beginning, intermediate and end parts of a chunk. If a token does not belong to any named entity, it is tagged as "O." Since IREX defined eight NE classes, tokens are classified into 33 (= 8 × 4 + 1) NE tags. For example, NE tags are assigned as following:

(1) *Kotoshi* (this year) *4 gatsu* (April) *Roma* (Rome) *ni* (to) *itta.* (went)
B-DATE I-DATE E-DATE S-LOCATION O O
(ϕ went to Rome on April this year.)

3 Motivation for Our Approach

Our NER system utilizes structural information. In this section, we describe the motivation for our approach.

High-performance Japanese NER systems are often based on supervised learning, and most of them use only local features, such as features obtained from the target token, two preceding tokens and two succeeding tokens. However, in some cases, NEs cannot be recognized by using only local features.

For example, while "*Kawasaki*" in the second sentence of (2) is the name of a person, "*Kawasaki*" in the second sentence of (3) is the name of a soccer team. However, the second sentences of (2) and (3) are exactly the same, and thus it is impossible to correctly distinguish these NE classes by only using information obtained from the second sentences.

(2) *Kachi-ha* (winner) *senpatsu-no* (starter) ***Kawasaki*** *Kenjiro.*
Kawasaki-ha genzai (now) *4 shou* (won) *3 pai.* (lost)
(The winning pitcher is the starter Kenjiro **Kawasaki**. Kawasaki has won 4 and lost 3.)

(3) *Dai* (the) *10 setsu-wa* (round) ***Kawasaki*** *Frontale-to taisen.* (against)
Kawasaki-ha genzai (now) *4 shou* (won) *3 pai.* (lost)
(The 10th round is against **Kawasaki** Frontale. Kawasaki has won 4 and lost 3.)

In order to recognize these NE classes, it is essential to use the information obtained from the previous context. Therefore, we utilize information obtained from the NER for the previous context: **cache feature** and **coreference relation**.

For another example, "*Shingishu*" in (4) is the name of city in North Korea. The most important clue for recognizing "*Shingishu*" as "LOCATION" may be the information obtained from the head verb, "*wataru* (get across)."

(4) *Shingishu-kara* (Sinuiju from) *Ouryokko-wo* (Amnokkang) ***wataru.*** (get across)
(ϕ **gets across** the Amnokkang River from Sinuiju.)

However, when using only local features, the word "*wataru*" is not taken into consideration because there are more than two morphemes between "*shu*[2]" and "*wataru*." In order to deal with such problem, we use the information obtained from the head verb: **syntactic feature** and **caseframe feature**.

4 NER Using Structural Information

4.1 Outline of Our NER System

Our NER system performs the chunking process based on morpheme units because character-based methods do not outperform morpheme-based methods (Masayuki and Matsumoto, 2003) and are not suitable for considering wider context.

A wide variety of trainable models have been applied to Japanese NER task, including maximum entropy models (Utsuro et al., 2002), support vector machines (Nakano and Hirai, 2004; Yamada, 2007) and conditional random fields (Fukuoka, 2006). Our system applies SVMs because, for Japanese NER, SVM-based systems achieved higher F-measure than the other systems. (Isozaki and Kazawa, 2003) proposed an SVM-based NER system with Viterbi search, which outperforms an SVM-based NER system with sequential determination, and our system basically follows this system. Our NER system consists of the following four steps:

1. Morphological analysis
2. Syntactic, case and coreference analyses
3. Feature extraction for chunking
4. SVM and Viterbi search based chunking

The following sections describe each of these steps in detail.

[2] Since the dictionary for morphological analysis has no entry "*Shingishu*," "*Shingishu*" is analyzed as consisting of three morphemes: "*shin*," "*gi*" and "*shu*."

Input sentence:

Gai	*mu*	*sho*	*no*	*shin*	*Bei*	*ha*	.
foreign	affairs	ministry	in	pro	America	group	

(Pro-America group in the Ministry of Foreign Affairs.)

Output of JUMAN:

Gaimu	*sho*	*no*	*shin*	*Bei*	*ha*	.
noun	noun	particle	noun	noun	noun	

Output of ChaSen:

Gaimusho	*no*	*shin-Bei*	*ha*	.
noun	particle	noun	noun	

Figure 1: Example of morphological analyses.

4.2 Morphological Analysis

While most existing Japanese NER systems use ChaSen (Matsumoto et al., 2003) as a morphological analyzer, our NER system uses a Japanese morphological analyzer JUMAN (Kurohashi and Kawahara, 2005) because of the following two reasons.

First, JUMAN tends to segment a sentence into smaller morphemes than ChaSen, and this is a good tendency for morpheme-based NER systems because the boundary contradictions between morphological analysis and NEs are considered to be reduced. Figure 1 shows an example of the outputs of JUMAN and ChaSen. Although both analyses are reasonable, JUMAN divided "*Gaimusho*" and "*shin-Bei*" into two morphemes, while ChaSen left them as a single morpheme. Second, JUMAN adds categories to some morphemes, which can be utilized for NER. In JUMAN, about thirty categories are defined and tagged to about one fifth of morphemes. For example, "*ringo* (apple)," "*inu* (dog)" and "*byoin* (hospital)" are tagged as "FOOD," "ANIMAL" and "FACILITY," respectively.

4.3 Syntactic, Case and Coreference Analyses

syntactic analysis Syntactic analysis is performed by using the Japanese parser KNP (Kurohashi and Nagao, 1994). KNP employs some heuristic rules to determine the head of a modifier.

case analysis Case analysis is performed by using the system proposed in (Kawahara and Kurohashi, 2002). This system uses Japanese case frames that are automatically constructed from a large corpus. To utilize case analysis for NER, we constructed case frames that include NE labels in advance. We explain details in Section 4.4.2. The case analysis is applied to each predicate in an input sentence. For details see (Kawahara and Kurohashi, 2002).

coreference analysis Coreference analysis is performed by using the coreference analyzer proposed by (Sasano et al., 2007). As will be mentioned in Section 4.4.2, our NER system uses coreference relations only when coreferential expressions do not share same morphemes. Basically, such coreference relations are recognized by using automatically acquired synonym knowledge.

4.4 Feature Extraction

4.4.1 Basic Features

As basic features for chunking, our NER system uses the morpheme itself, character type, POS tag and category if it exists.

As character types, we defined seven types: "*kanji*," "*hiragana*," "*katakana*," "*kanji* with *hiragana*," "punctuation mark," "alphabet" and "digit." As for POS tag, more than one POS feature are extracted if the target morpheme has POS ambiguity. In addition, besides POS tag obtained by JUMAN, our system also uses POS tag obtained from Japanese morphological analyzer MeCab[3] that uses IPADIC as a word dictionary (Asahara and Matsumoto, 2002). The JUMAN dictionary has few named entity entries; thus our system supplements the lack of lexical knowledge by using MeCab.

4.4.2 Structural Features

Our NER system uses three types of global features: cache features, syntactic features and caseframe features, and a rule that reflects coreference relations. Although the coreference relations are not used as features, we describe how to use them in this section.

cache feature If the same morpheme appears multiple times in a single document, in most cases the NE tags of these morphemes have some relation to each other, and the NER results for previous parts of the document can be a clue for the analysis for following parts.

We consider the examples (2) and (3) again. Although the second sentences of (2) and (3) are exactly the same, we can recognize "*Kawasaki*" in the second sentence of (2) is "S-PERSON" and "*Kawasaki*" in the second sentence of (3) is "S-ORGANIZATION" by reading the first sentences.

To utilize the information obtained from previous parts of the document, our system uses the NER results for previous parts of the document as features, called cache features. When analyzing (2), our system uses the outputs of NE recognizer for

[3]http://mecab.sourceforge.jp/

"*Kawasaki*" in the first sentence as a feature for "*Kawasaki*" in the second sentence. For simplicity, our system uses correct NE tags when training. That is, as a feature for "*Kawasaki*" in the second sentence of (2), the correct feature "B-PERSON" is always added when training, not always added when analyzing.

coreference rule Coreference relation can be a clue for NER. This clue is considered by using cache features to a certain extent. However, if the same morpheme is not used, cache features cannot work.

For example, "*NHK kokyo gakudan*" and "*N-kyo*" in (5) have coreference relation, but they do not share the same morpheme.

(5) ***NHK kokyo gakudan****-no ongaku kantoku-ni*
symphony orchestra musical director
shuunin. *N-kyo-to kyoen-shite* *irai ... .*
became perform together since
(He became musical director of the **NHK Symphony Orchestra**. Since performing together with N-kyo ... .)

In this case, "*NHK kokyo gakudan*" can easily be recognized as "ORGANIZATION," because it ends with "*kokyo gakudan* (symphony orchestra)." Meanwhile, "*N-kyo*," the abbreviation of "*NHK kokyo gakudan*," cannot easily be recognized as "ORGANIZATION."

Therefore, our system uses a heuristic rule that if a morpheme sequence is analyzed to be coreferential to a previous morpheme sequence that is recognized as an NE class, the latter morpheme sequence is recognized as the same NE class. Since this heuristic rule is introduced in order to utilize the coreference relation that is not reflected by cache features, our system applies this rule only when coreferential expressions do not have any morphemes in common.

syntactic feature As mentioned in Section 3, our system utilizes the information obtained from the head verb. As syntactic features, our system uses the head verb itself and the surface case of the *bunsetsu* that includes the target morpheme.

For the morpheme "*shin*" in example (4), the head verb "*wataru* (get across)" and the surface case "*kara* (from)" are added as syntactic features.

caseframe feature Syntactic features cannot work if the head verb does not appear in the training data. To overcome this data sparseness problem, caseframe features are introduced.

Table 2: Case frame of "*haken* (dispatch)."

case	examples
ga (nominative)	Japan:23,party:13,country:12,government:7, company6,ward:6,corps:5,UN:4,US:4,Korea:4, team:4,. . . (ORGANIZATION,LOCATION)
wo (objective)	party:1249,him:1017,soldier:932,official:906, company6:214,instructor:823,expert:799, helper:694,staff:398,army:347,. . .
ni (locative)	Iraq:700,on-the-scene:576,abroad:335, home:172,Japan:171,Indirect Ocean:142, scene:141,China:125,. . . (LOCATION)

For example, although the head verb "*haken* (dispatch)" can be a clue for recognizing "*ICAO*" in (6) as "ORGANIZATION," syntactic features cannot work if "*haken* (dispatch)" did not appear in the training data.

(6) *ICAO-ha genchi-ni senmonka-wo* ***haken****-shita.*
scene to expert dispatched
(*ICAO* **dispatched** experts to the scene)

However, this clue can be utilized if there is knowledge that the "*ga* (nominative)" case of "*haken* (dispatch)" is often assigned by "ORGANIZATION."

Therefore, we construct case frames that include NE labels in advance. Case frames describe what kinds of cases each verb has and what kinds of nouns can fill a case slot. We construct them from about five hundred million sentences. We first recognize NEs appearing in the sentences by using a primitive NER system that uses only local features, and then construct the case frames from the NE-recognized sentences. To be more specific, if one tenth of the examples of a case are classified as a certain NE class, the corresponding label is attached to the case. Table 2 shows the constructed case frame of "*haken* (dispatch)." In the "*ga* (nominative)" case, the NE labels, "ORGANIZATION" and "LOCATION" are attached.

We then explain how to utilize these case frames. Our system first performs case analysis, and uses as caseframe features the NE labels attached in the case to which the target morpheme is assigned. For instance, by the case analyzer, the postpositional particle "*-ha*" in (6) is recognized as meaning nominative and "*ICAO*" is assigned to the "*ga* (nominative)" case of the case frame of "*haken* (dispatch)."Therefore, the caseframe features, "ORGANIZATION" and "LOCATION" are added to the features for the morpheme "*ICAO*."

4.5 SVM and Viterbi Search Based Chunking

To utilize cache features obtained from the previous parts of the same sentence, our system determines

Table 3: Experimental results (F-measure).

	CRL	IREX	WEB
baseline	88.63	85.47	68.98
+ cache	88.81 +0.18*	85.94 +0.47	69.67 +0.69*
+ coreference	88.68 +0.05	86.52 +1.05***	69.17 +0.19
+ syntactic	88.80 +0.17*	85.77 +0.30	70.25 +1.27**
+ caseframe	88.57 −0.06	85.51 +0.04	70.12 +1.14*
+ thesaurus	88.77 +0.14	86.36 +0.89*	68.63 −0.35
use all	**89.40** +0.77***	**87.72** +2.25***	**71.03** +2.05***

significant at the .1 level:*, .01 level:**, .001 level:***

NE tags clause by clause. The features extracted from two preceding morphemes and two succeeding morphemes are also used for chunking a target morpheme. Since SVM can solve only a two-class problem, we have to extend a binary classifier SVM to n-class classifier. Here, we employ the one versus rest method, in which we prepared n binary classifiers and each classifier is trained to distinguish a class from the rest of the classes.

To consider consistency of NE tags in a clause, our system uses Viterbi search with some constraints such as a "B-DATE" must be followed by "I-DATE" or "E-DATE." Since SVMs do not output probabilities, our system uses the SVM+sigmoid method (Platt et al., 2000). That is, a sigmoid function $s(x) = 1/(1+exp(-\beta x))$ is applied to map the output of SVM to a probability-like value. Our system determines NE tags by using these probability-like values. Our system is trained by TinySVM-0.09[4] with C = 0.1 and uses a fixed value β = 10. This process is almost the same as the process proposed by Isozaki and Kazawa and for details see (Isozaki and Kazawa, 2003).

5 Experiments

5.1 Data

For training, we use CRL NE data, which was prepared for IREX. CRL NE data has 18,677 NEs on 1,174 articles in Mainichi Newspaper.

For evaluation, we use three data: CRL NE data, IREX's formal test data called GENERAL and WEB NE data. When using CRL NE data for evaluation, we perform five-fold cross-validation. IREX test data has 1,510 NEs in 71 articles from Mainichi Newspaper. Although both CRL NE data and IREX test data use Mainichi Newspaper, these formats are not the same. For example, CRL NE data removes parenthesis expressions, but IREX test data does not. WEB NE data, which we annotated NEs on corpus collected from the Web, has 1,686 NEs in 354 articles. Although the domain of the web corpus differs from that of CRL NE data, the format of the web corpus is the same as CRL NE data format.

[4]http://chasen.org/ taku/software/TinySVM/

5.2 Experiments and Discussion

To confirm the effect of each feature, we conducted experiments on seven conditions as follows:

1. Use only basic features (baseline)
2. Add cache features to baseline
3. Add the coreference rule to baseline
4. Add parent features to baseline
5. Add caseframe features to baseline
6. Add thesaurus features to baseline
7. Use all structural information and thesaurus

Since (Masayuki and Matsumoto, 2003; Nakano and Hirai, 2004) reported the performance of NER system was improved by using a thesaurus, we also conducted experiment in which semantic classes obtained from a Japanese thesaurus "*Bunrui Goi Hyo*" (NLRI, 1993) were added to the SVM features. Table 3 shows the experimental results.

To judge the statistical significance of the differences between the performance of the baseline system and that of the others, we conducted a McNemar-like test. First, we extract the outputs that differ between the baseline method and the target method. Then, we count the number of the outputs that only baseline method is correct and that only target method is correct. Here, we assume that these outputs have the binomial distribution and apply binomial test. As significance level, we use .1 level, .01 level and .001 level. The results of the significance tests are also shown in Table 3.

When comparing the performance between data sets, we can say that the performance for WEB NE data is much worse than the others. This may be because the domain of the WEB corpus differs from that of CRL NE data.

As for the differences in the same data set, cache features and syntactic features improve the performance not dramatically but consistently and independently from the data set. The coreference rule also improves the performance for all data sets, but especially for IREX test data. This may be because IREX test data does not remove parenthesis expressions, and thus there are a many coreferential expressions in the data. Caseframe features improve the performance for WEB NE data, but do not contribute to the performance for CRL NE data and

Table 4: Comparison with previous work.

	CRL cross validation	IREX test data	Learning Method	Analysis Units	Features
(Isozaki and Kazawa, 2003)	86.77	85.10	SVM + Viterbi	morpheme	basic features
(Masayuki and Matsumoto, 2003)	87.21		SVM	character	+thesaurus
(Fukuoka, 2006)	87.71		Semi-Markov CRF	character	basic features
(Yamada, 2007)	88.33		SVM + Shift-Reduce	morpheme	+bunsetsu features
(Nakano and Hirai, 2004)	89.03		SVM	character	+bunsetsu features & thesaurus
Our system	**89.40**	**87.72**	SVM + Viterbi	morpheme	+structural information & thesaurus

IREX test data. This result shows that caseframe features are very generalized features and effective for data of different domain. On the other hand, thesaurus features improve the performance for CRL NE data and IREX test data, but worsen the performance for WEB NE data. The main cause for this may be overfitting to the domain of the training data.

By using all structural information, the performance is significantly improved for all data sets, and thus we can say that the structural information improves the performance of NER.

5.3 Comparison with Previous Work

Table 4 shows the comparison with previous work for CRL NE data and IREX test data. Our system outperforms all other systems, and thus we can confirm the effectiveness of our approach.

6 Conclusion

In this paper, we presented an approach that uses structural information for Japanese NER. We introduced four types of structural information to an SVM-based NER system: cache features, coreference relations, syntactic features and caseframe features, and conducted NER experiments on three data. As a consequence, the performance of NER was improved by using structural information and our approach achieved a higher F-measure than existing approaches.

References

Masayuki Asahara and Yuji Matsumoto, 2002. *IPADIC User Manual*. Nara Institute of Science and Technology, Japan.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *Proc. of COLING 2002*, pages 1–7.

Kenta Fukuoka. 2006. Named entity extraction with semi-markov conditional random fields (in Japanese). Master's thesis, Nara Institute of Science and Technology.

IREX Committee, editor. 1999. *Proc. of the IREX Workshop*.

Hideki Isozaki and Hideto Kazawa. 2003. Speeding up support vector machines for named entity recognition (in japanese). *Trans. of Information Processing Society of Japan*, 44(3):970–979.

Heng Ji and Ralph Grishman. 2005. Improving name tagging by reference resolution and relation detection. In *Proc. of ACL-2005*, pages 411–418.

Daisuke Kawahara and Sadao Kurohashi. 2002. Fertilization of Case Frame Dictionary for Robust Japanese Case Analysis. In *Proc. of COLING-2002*, pages 425–431.

Sadao Kurohashi and Daisuke Kawahara. 2005. Japanese morphological analysis system JUMAN version 5.1 manual.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

R. Malouf. 2002. Markov models for language-independent named entity recognition. In *Proc. of CoNLL-2002*, pages 187–190.

Asahara Masayuki and Yuji Matsumoto. 2003. Japanese named entity extraction with redundant morphological analysis. In *Proc. of HLT-NAACL 2003*, pages 8–15.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka, and Masayuki Asahara. 2003. Morphological analysis System chasen 2.3.3 users manual.

Behrang Mohit and Rebecca Hwa. 2005. Syntax-based semi-supervised named entity tagging. In *Proc. of ACL Interactive Poster and Demonstration Sessoins*, pages 57–60.

MUC-6. 1995. *Proc. of the Sixth Message Understanding Conference*. Morgan Kaufmann Publishers, INC.

Keigo Nakano and Yuzo Hirai. 2004. Japanese named entity extraction with bunsetsu features (in Japanese). *Trans. of Information Processing Society of Japan*, 45(3):934–941.

The National Language Institute for Japanese Language, NLRI, editor. 1993. *Bunrui Goi Hyo (in Japanese)*. Shuuei Publishing.

John C. Platt, Nello Cristiani, and John ShaweTaylor. 2000. Lage margin DAGs for multiclas classification. In *Advances in Neural Information Processing System 12*.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2007. Improving coreference resolution using bridging reference resolution and automatically acquired synonyms. In *Proc. of DAARC-2007*.

Takehito Utsuro, Manabu Sassano, and Kiyotaka Uchimoto. 2002. Combing outputs of multiple named entity chunkers by stacking. In *Proc. of EMNLP-2002*.

Hiroyasu Yamada. 2007. Shift reduce chunking for Japanese named entity extraction (in Japanese). In *IPSJ SIG Notes NL-179-3*, pages 13–18.

Dimensionality Reduction with Multilingual Resource

YingJu Xia **Hao Yu** **Gang Zou**
Fujitsu Research & Development Center Co.,LTD.
13F Tower A, Ocean International Center, No.56 Dong Si Huan Zhong Rd, Chaoyang District, Beijing, China, 100025
{yjxia,yu,zougang}@cn.fujitsu.com

Abstract

Query and document representation is a key problem for information retrieval and filtering. The vector space model (VSM) has been widely used in this domain. But the VSM suffers from high dimensionality. The vectors built from documents always have high dimensionality and contain too much noise. In this paper, we present a novel method that reduces the dimensionality using multilingual resource. We introduce a new metric called *TC* to measure the term consistency constraints. We deduce a *TC* matrix from the multilingual corpus and then use this matrix together with the term-by-document matrix to do the Latent Semantic Indexing (LSI). By adopting different *TC* threshold, we can truncate the *TC* matrix into small size and thus lower the computational cost of LSI. The experimental results show that this dimensionality reduction method improves the retrieval performance significantly.

1 Introduction

1.1 Basic concepts

The vast amount of electronic information that is available today requires effective techniques for accessing relevant information from it. The methodologies developed in information retrieval aim at devising effective means to extract relevant documents in a collection when a user query is given. In information retrieval and filtering, Query and document representation is a key problem and many techniques have been developed. Among these techniques, the vector space model (VSM) proposed by Salton (1971; 1983) has been widely used. In the VSM, a document is represented by a vector of terms. The cosine of the angle between two document vectors indicates the similarity between the corresponding documents. A smaller angle corresponds to a larger cosine value and indicates higher document similarity. A query, which describes the information need, is encoded as a vector as well. Retrieval of documents that satisfy the information need is achieved by finding the documents most similar to the query, or equivalently, the document vectors closest to the query vector. There are several advantages to this approach beyond its mathematical simplicity. Above all, it is efficient to compute and store the word counts. This is one reason that why VSM is widely used for query and document representation. But this method has problem that the vectors built from documents always have high dimensionality and contain too much noise. The high dimensionality causes high computational and memory requirements while noise in the vectors degrades the system performance.

1.2 Related works

To address these problems, many dimensionality reduction techniques have been applied to query and document representation. Among these techniques, Latent Semantic Indexing (LSI) (Deerwester et al., 1990; Hofmann, 1999; Ding, 2000; Jiang and Littman, 2000; Ando, 2001; Kokiopoulou and Saad, 2004; Lee et al., 2006) is a well-known approach. LSI constructs a smaller document matrix that retains only the most important information from the original by using the Singular Value Decomposition (SVD). Many modifications have been made to this approach (Hofmann, 1999; Ding, 2000; Jiang and Littman, 2000; Kokiopoulou

and Saad, 2004; Sun et al., 2004; Husbands et al., 2005). Among them, IRR (Ando and Lee, 2001) is a subspace-projection method that counteracts tendency to ignore minority-class documents. This is done by repeatedly rescaling vectors to amplify the presence of documents poorly represented in previous iterations.

In concept indexing (CI) (Karypis and Han, 2000) method, the original set of documents is first clustered into *k* similar groups, and then for each group, the centroid vector (i.e., the vector obtained by averaging the documents in the group) is used as one of the *k* axes of the lower dimensional space. The key motivation behind this dimensionality reduction approach is the view that each centroid vector represents a concept present in the collection, and the lower dimensional representation expresses each document as a function of these concepts. George and Han (2000) extend concept indexing in the context of supervised dimensionality reduction. To capture the concept, phrase also has been used as indexing entries (Mao and Chu, 2002).

The LPI method (Isbell and Viola, 1999) tries to discover the local structure and obtains a compact document representation subspace that best detects the essential semantic structure. The LPI uses Locality Preserving Projections (LPP) (Xiaofei He and Partha, 2003) to learn a semantic space for document representation. Xiaofei He et al., (2004) try to get sets of highly-related words, queries and documents are represented by their distance to these sets. These algorithms have successfully reduced the dimensionality and improve the retrieval performance but at the mean time they led to a high computational complexity.

1.3 Our method

In this study, we propose a novel method that reduces the dimensionality using multilingual resource. We first introduce a new metric called *TC* to measure the term consistency constraints. We use this metric to deduce a *TC* matrix from the multilingual corpus. Then we combine this matrix to the term-by-document matrix and do the Latent Semantic Indexing. By adopting different *TC* threshold, we can truncate the *TC* matrix into small size and thus lower the computational cost of LSI.

The remainder of this paper is organized as follows. Section 2 describes the dimensionality reduction method using multilingual resource. Section 3 shows the experimental results to evaluate the dimensionality reduction method. Finally, we provide conclusions and remarks of future work in Section 4.

2 Dimensionality reduction using multilingual resource

2.1 Motivation

As mentioned above, the queries and documents are represented by vectors of terms. The weight of each term indicates its contribution to the vectors. Many weighting schemes have been proposed. The simplest form is to use the term-frequency (TF) as the term weight. In this condition, a document can be represented as a vector $\vec{d} = (tf_1, tf_2, ..., tf_n)$, where tf_i is the frequency of the *i*th term in the document. A widely used refinement to this model is to weight each term based on its inverse document frequency (IDF) in the documents collection. This is commonly done by multiplying the frequency of each term i by $\log(N / df_i)$, where N is the total number of documents in the collection, and df_i is the number of documents that contain the *i*th term. This leads to the TF-IDF representation of the documents. Although the TF-IDF weighting scheme has many variants (Buckley, 1985; Berry et al., 1999; Robertson et al., 1999), the idea is the same one that uses the statistical information such as TF and IDF to calculate the term weight of vectors.

This kind of statistical information is independence with languages. For example, in one language, say L^a, we have a vocabulary $V^a = \{w_1^a, w_2^a, ..., w_n^a\}$ and a documents collection $D^a = \{d_1^a, d_2^a, ..., d_m^a\}$. If this documents collection has a parallel corpus in language L^b, say, $D^b = \{d_1^b, d_2^b, ..., d_m^b\}$ and a vocabulary $V^b = \{w_1^b, w_2^b, ..., w_n^b\}$. When we put a query $Q_k^a = \{q_{k1}^a, q_{k2}^a, ..., q_{kl}^a\}$ ($q_{ki}^a \in V^a$) into an information retrieval system. The information retrieval system will converts the query Q_k^a and the documents in the collection D^a into vectors. By calculating the similarity between query Q_k^a and each document d_i^a, the system selects the documents whose similarity is higher than a threshold as the results R_k^a. If we translate the query Q_k^a into language L^b and get query Q_k^b, when putting the Q_k^b into the same information retrieval system, we get the retrieval results R_k^b. Since the Q_k^a and Q_k^b contain the same content and only expressed in different languages. We expect that R_k^a

and R_k^b will contain the same content. If this assumption holds, the vocabulary which is used to build queries and documents vectors should have high representative ability. Since the weight of each term in the vector is calculated by the statistical information such as TF and IDF. If the vocabulary V^a and V^b have high representative ability, their statistical information will be consistent as well. This is the main motivation of our dimensionality reduction method.

2.2 Dimensionality reduction method

The most straightforward way to measure the word's representability in multilingual resource is to calculate the TF and IDF of each word in different languages. But this method has one problem that the TF-IDF scheme is dedicated for each single document, the same word will have different weight in different documents. It is impractical to impose the consistency constraint to every document. Even we can do that, this method still has the drawback that it is very difficult to port to another documents collection. To address this problem, we consider the whole documents collection as one single document. In this condition, the IDF will be a fixed number.

We introduce a new metric to measure the term consistency called *TC*. Figure 1 and Figure 2 illustrate the basic idea. In these figures, the curve L_a shows the word logarithmic frequency in the documents collection of language L^a, the curve L_b shows the corresponding translation's logarithmic frequency in the documents collection of language L^b. TC_i and TC_j are the term consistency of w_i and w_j respectively.

Figure 1 shows the *TC* in normal condition that the average word frequency in language *a* is proximate to that of language *b*. In this case, the *TC* is defined as below:

$$TC(w_i^b) = \min(\log(f_i^a)/\log(f_i^b), \log(f_i^b)/\log(f_i^a)) \quad (1)$$

Here f_i^a is the frequency of w_i^a in language *a*. f_i^b is the frequency of the w_i^a's translation in language *b*. In multilingual case, the $TC(w_i)$ will be defined as below:

$$TC(w_i) = \min(TC(w_i^b), \ldots TC(w_i^n)) \quad (2)$$

In the case that the average word frequency in language *a* is different with that of language *b*, we will first calculate the moving average as shown in the Figure 2. After that, we use the moving average to calculate the *TC* of w_i as below:

$$TC(w_i^b) = \min((\log(f_i^a) + H)/\log(f_i^b), \log(f_i^b)/(\log(f_i^a) + H)) \quad (3)$$

Here *H* is distance between the moving average and the original one.

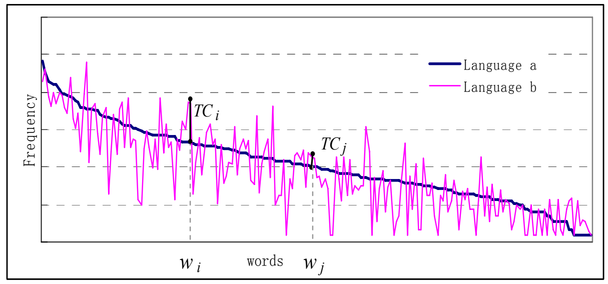


Figure 1. *TC* in normal condition

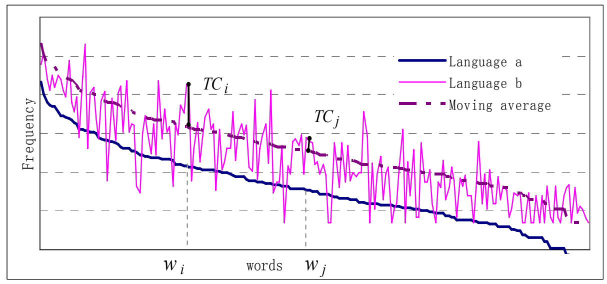


Figure 2. *TC* in shift condition

Once we get the *TC* of every word in language *a*, we present it in a diagonal matrix $T_{t\times t} = diag(TC_1, TC_2, \ldots, TC_t)$, $TC_1 \geqslant TC_2 \geqslant \ldots \geqslant TC_t$.

When applying the *TC* matrix $T_{t\times t}$ in information retrieval, we combine $T_{t\times t}$ into the term-by-document matrix $A_{t\times d}$. Where $A_{t\times d} = [a_{ij}]$ and the a_{ij} is the weight of term *i* in document *j*. We get a new matrix $B_{t\times d} = T_{t\times t}A_{t\times d}$. Then following the classical LSI, we replace $B_{t\times d}$ by a low-rank approximation derived from its truncated Singular Value Decomposition (SVD):

$$B_{t\times d} = U_{t\times n}\Sigma_{n\times n}V_{d\times n}^T$$

Here $UU^T = I$, $VV^T = I$, $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_n)$ $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq \sigma_{r+1} = \ldots = \sigma_n = 0$.

The main problem of LSI is that it usually led to a high computational complexity since the matrix $B_{t\times d}$ usually in 10^3-10^5 dimensional space. To lower the computational cost, we truncate the *TC* matrix $T_{t\times t}$ according to different *TC* threshold and get a new matrix $\hat{T}_{t\times t} = diag(TC_1, TC_2, \ldots, TC_t)$, $TC_1 \geq TC_2 \geq \ldots \geq TC_r \geq TC_{r+1} = \ldots = TC_t = 0$. Then we get $\hat{B}_{r\times d} = \hat{T}_{r\times r}A_{r\times d}$. Since *r* is small than *t*, the

computational cost on the matrix $\hat{B}_{r\times d}$ will lower than $B_{t\times d}$. Note that the matrix $\hat{B}_{r\times d}$ is deduced from the *TC* matrix $T_{t\times t}$ which is sorted by word representative ability. It will contain less noise and outperform the original matrix $A_{t\times d}$. The experimental results have shown the effective of this method.

For one word w_i^a in language L^a, there are always several translations in language L^b, say (w_{i1}^b, w_{i2}^b,..., w_{ik}^b). To handle this one-to-many phenomenon, we calculate the co-occurrence of w_i^a and each translation and select the highest one as the translation of w_i^a.

3 Experiments

We adopt a VSM based IR system to evaluate the dimensionality reduction method presented in Section 2. The term weight in the term-by-document matrix is calculated by the TF-IDF weighting scheme.

3.1 Training and test corpora

The training corpus comes from Chinese Linguistic Data Consortium (http://www.chineseldc.org/, abbreviate as CLDC). Its code number is "2004-863-009". This parallel corpus contains parallel texts in Chinese, English and Japanese. It is aligned to sentence level. The sentence alignment is manually verified and the sampling examination shows the accuracy reaches 99.9%.

The experiments are conducted on two test corpora. The first one is the information retrieval test corpus gotten from CLDC ("2003-863-006"). It is a Chinese IR corpus and contains 20 topics for test. Each topic has key words and description and narrative. The second one is the Reuters 2001 data (http://about.reuters.com/researchandstandards/corpus/). This corpus is a collection of about 810,000 Reuters English news stories from August 20, 1996 to August 19, 1997. It was used by the TREC-10 Filtering Tracks (Robertson and Soboroff, 2002). In TREC-10, 84 Reuters categories were used to simulate user profiles.

The evaluate measure is a version of van Rijsbergen(1979)'s *F* measure with β=1(we denote it as *F1*).

3.2 Experimental results

The table1 and table2 show the experimental results conducted on Chinese and English test Corpus respectively. In these tables, we compare our method with basic LSI and LPI (Xiaofei et.al, 2004). In the table1, the 'C-E' means the *TC* matrix gotten from Chinese-English training collection (deduced from the trilingual training corpus). The 'C-J' means that the *TC* matrix gotten from Chinese-Japanese training collection, and so force the 'C-E-J'. All the *TC* matrices have been normalized to range from 0 to 1. The threshold θ is used to truncate the *TC* matrix into small size. Bigger θ corresponds to smaller truncated *TC* matrix. Note that here θ is discrete since for some θ, the size of truncated matrix is very similar. For example, when θ = 0.85 and θ = 0.9, the size of truncated *TC* matrices are the same one.

LSI: 0.3785, LPI: 0.405			
θ	C-E	C-J	C-E-J
0.3	0.404	0.4014	0.4124
0.4	0.4098	0.406	0.4185
0.45	0.4159	0.4185	0.4226
0.5	0.4204	0.4124	0.4105
0.55	0.4061	0.4027	0.3997
0.6	0.3913	0.3992	0.396
0.8	0.3856	0.3867	0.3842
0.85	0.3744	0.3754	0.3768

Table1.*F1* measure of Chinese test corpus

LSI: 0.3416, LPI: 0.3556			
θ	E-C	E-J	E-C-J
0.3	0.356	0.3478	0.3578
0.4	0.3578	0.3596	0.3702
0.45	0.3698	0.3651	0.3734
0.5	0.3636	0.3575	0.363
0.55	0.3523	0.3564	0.3477
0.6	0.3422	0.3448	0.3458
0.8	0.3406	0.3397	0.3378
0.85	0.3304	0.3261	0.3278

Table2. *F1* measure of English test corpus

From the experimental results, we can see that our method make great enhancement to the basic LSI method. And our method also outperforms the LPI method in both test corpora. Comparing the performance on different training collection, we can find that the difference is subtle. In Chinese test corpus, the *TC* matrix gotten from C-E-J training collection get the best performance (*F1*=0.4226) at θ=0.45 while the C-E test collection get 0.4204

at θ=0.5 and the C-J test collection get 0.4185 at θ=0.45. For the English test corpus, the trilingual training collection also gets the best performance. But the difference between bilingual and trilingual training collection is also subtle (E-C-J: *F1*=0.3734, E-C: *F1*=0.3698, E-J: *F1*=0.3651). In the English test corpus, all the training collection get the best performance at θ=0.45.

As mentioned before, the bigger θ means the smaller size of the truncated *TC* matrix. While small size of the truncated *TC* matrix means low computational cost and high system speed. This is one of the advantages of our method over the traditional LSI method. We conducted some experiments to test the system speed on different threshold θ. We use the number of documents per second (docs/s) to denote this kind of system speed. The experiment is conducted on the personal computer with a Pentium (R) 4 processor @2.8GHz, 256 KB cache and 512 MB memory. Table 3 shows the experimental results that the θ vs. system speed and Figure 3 illustrates the *F1* measure vs. the system speed.

Baseline(LSI): 566.5 docs/s			
θ	C-E	C-J	C-E-J
0.3	1039.3	1034.4	1355.0
0.4	1148.4	1188.9	1372.5
0.45	1290.5	1246.9	1391.3
0.5	1323.9	1323.3	1469.6
0.55	1393.3	1392.6	1563.8
0.6	1413.3	1508.8	1590.1
0.8	1513.1	1555.6	1660.5
0.85	1641.1	1778.2	1773.5

Table 3. θ vs. system speed

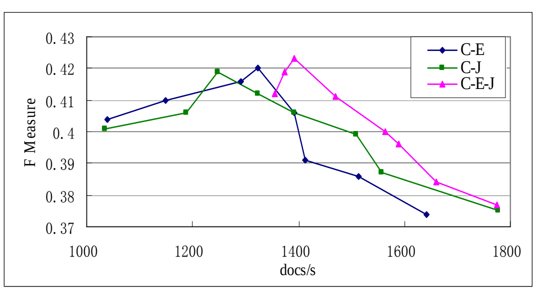


Figure 3. *F1* measure vs. system speed

4 Conclusions

In this paper, we present a novel method that reduces the dimensionality using multilingual resource. We deduce a *TC* matrix from the multilingual corpus and then truncate it to small size according to different *TC* threshold. Then we use the truncated matrix together with the term-by-document matrix to do the LSI analysis. Since the truncated *TC* matrix is sorted by word representative ability. It will contain less noise than the original term-by-document matrix. The experimental results have shown the effectiveness of this method.

In the future, we will try to find the optimal truncate threshold θ automatically. And since it is more difficult to get the parallel corpora than comparable corpora, we will explore using comparable corpora to do the dimensionality reduction.

Acknowledgement

This research was carried out through financial support provided under the NEDO International Joint Research Grant Program (NEDO Grant).

References

Ando R. K., “Latent Semantic Space: Iterative Scaling improves precision of inter-document similarity measurement”, in Proc. of the 23th International ACM SIGIR, Athens, Greece, 2000.

Ando R. K., and Lee L., “Iterative Residual Rescaling: An Analysis and Generalization of LSI”, in Proc. of the 24th International ACM SIGIR, New Orleans, LA, 2001.

Arampatzis A., Beney J., Koster C.H.A., and T.P. van der Weide. KUN on the TREC9 Filtering Track: Incrementality, decay, and theshold optimization for adaptive filtering systems. The ninth Text Retrieval Conference, November 9-12, 2000 Gaithersburg, MD,

Avi Arampatzis and Andre van Hameren The Score-Distributional Threshold Optimization for Adaptive Binary Classification Tasks , SIGIR’01, September 9-12,2001, New Orleans, Louisiana,USA. 285-293

Berry M., Drmac Z., and Jessup E.. Matrices, vector spaces, and information retrieval. SIAM Review, 41(2):pp335-362, 1999.

Bingham E. and Mannila H., “Random Projection in dimensionality reduction: applications to image and text data”, Proc. Of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 245-250,2001.

Buckley C.. Implementation of the SMART information retrieval system. Technical Report TR85-686, Department of Computer Science, Cornell University,

Ithaca, NY 14853, May 1985. Source code available at ftp://ftp.cs.cornell.edu/pub/smart.

C.H. Lee, H.C. Yang, and S.M. Ma, "A Novel Multi-Language Text Categorization System Using Latent Semantic Indexing", The First International Conference on Innovative Computing, Information and Control (ICICIC-06), Beijing, China, 2006.

C.J. van Rijsbergen. Information Retrieval, chapter 7. Butterworths, 2 edition, 1979.

Deerwester S. C., Dumais S. T., Landauer T. K., Furnas G. W., and harshman R. A., "Indexing by Latent Semantic Analysis", Journal of the American Society of Information Science, 41(6):391-407, 1990.

Ding C. H.. A probabilistic model for dimensionality reduction in information retrieval and filtering. In Proc. of 1st SIAM Computational Information Retrieval Workshop, October 2000.

George Karypis, Eui-Hong (Sam) Han, Fast supervise d dimensionality reduction algorithm with applications to document categorization & retrieval ,Proceedings of the ninth international conference on Information and knowledge management, November 2000

Hofmann T., "Probabilistic Latent Semantic Indexing", in Proc. of the 22th International ACM SIGIR, Berkeley, California, 1999.

Husbands, P., Simon, H., and Ding, C. Term norm distribution and its effects on latent semantic indexing, Information Processing and Management: an International Journal, v.41 n.4, p.777-787, July 2005

Isbell C. L. and Viola P., "Restructuring Sparse High Dimensional Data for Effective Retrieval", Advances in Neural Information Systems, 1999.

Jiang F. and Littman M.L., Approximate dimension equalization in vector-based information retrieval. Proc. 17th Int'l Conf. Machine Learning, 2000.

Karypis G. and Han E.H.. Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval & categorization. Technical Report TR-00-016, Department of Computer Science, University of USA

Kokiopoulou E., Saad Y., Polynomial filtering in latent semantic indexing for information retrieval ,Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR '04,July 2004

Mao W. and Chu W.W.. Free-text medical document retrieval via phrase-based vector space model. In Proceedings of AMIA Annual Symp 2002.

Minnesota, Minneapolis, 2000. Available on the WWW at URL http://www.cs.umn.edu/~karypis.

Robertson SE, Walker S, Beaulieu M,Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track- Proceedings of the seventh Text Retrieval Conference, TREC-7, pp. 253-264 ,1999

Robertson, S., & Soboroff, I., The TREC-10 Filtering track final report. Proceeding of the Tenth Text REtrieval Conference (TREC-10) pp. 26-37. National Institute of Standards and Technology, special publication 500-250., 2002

Salton, G, the SMART Retrieval System – Experiments in Automatic Document Processing. Prentice-Hall, Englewood. Cliffs, New Jersey,1971.

Salton, G., Dynamic Information and Library processing. Prentice-Hall, Englewood Cliffs, New Jersey,1983.

Salton, G and McGill. M.J., Introduction to Modern Information retrieval. McGraw Hill, New York,1983.

Sun, J.T. , Chen , Z. , Zeng , H.J. , Lu, Y.C. , Shi, C.Y. and Ma, W.Y. ,"Supervised Latent Semantic Indexing for Document Categorization" , In Proceedings of the Fourth IEEE International Conference on Data Mining 2004

Xiaofei He and Partha Niyogi, "Locality Preserving Projections",in Advances in Neural Information Processing Systems 16, Vancouver, Canada, 2003.

Xiaofei He, Deng Cai, Haifeng Liu, Wei-Ying Ma, Locality preserving indexing for document representation, Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR '04, July 2004

Zhai C., Jansen P., Roma N., Stoica E., and Evans D.A.. Optimization in CLARIT adaptive filtering. In proceeding of the Eight Text Retrieval Conference 1999, 253-258.

Zhang Y., and Callan J.. Maximum likelihood Estimation for Filtering Thresholds. SIGIR'01, September 9-12,2001, New Orleans, Louisiana,USA. 294-302

A Web-based English Proofing System for English as a Second Language Users

Xing Yi[1], **Jianfeng Gao**[2] **and William B. Dolan**[2]
[1]Center for Intelligent Information Retrieval, Department of Computer Science
University of Massachusetts, Amherst, MA 01003-4610, USA
yixing@cs.umass.edu
[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{jfgao,billdol}@microsoft.com

Abstract

We describe an algorithm that relies on web frequency counts to identify and correct writing errors made by non-native writers of English. Evaluation of the system on a real-world ESL corpus showed very promising performance on the very difficult problem of critiquing English determiner use: 62% precision and 41% recall, with a false flag rate of only 2% (compared to a random-guessing baseline of 5% precision, 7% recall, and more than 80% false flag rate). Performance on collocation errors was less good, suggesting that a web-based approach should be combined with local linguistic resources to achieve both effectiveness and efficiency.

1 Introduction

Proofing technology for native speakers of English has been a focus of work for decades, and some tools like spell checkers and grammar checkers have become standard features of document processing software products. However, designing an English proofing system for English as a Second Language (ESL) users presents a major challenge: ESL writing errors vary greatly among users with different language backgrounds and proficiency levels. Recent work by Brockett *et al.* (2006) utilized phrasal Statistical Machine Translation (SMT) techniques to correct ESL writing errors and demonstrated that this data-intensive SMT approach is very promising, but they also pointed out SMT approach relies on the availability of large amount of training data. The expense and difficulty of collecting large quantities of raw and edited ESL prose pose an obstacle to this approach.

Search Phrase	Google.com	Live.com	Yahoo.com
English as Second Language	306,000	52,407	386,000
English as ***a*** Second Language	1,490,000	38,336,308	4,250,000

Table 1: Web Hits for Phrasal Usages

In this work we consider the prospect of using the Web, with its billions of web pages, as a data source with the potential to aid ESL writers. Our research is motivated by the observation that ESL users already use the Web as a corpus of good English, often using search engines to decide whether a particular spelling, phrase, or syntactic construction is consistent with usage found on the Web. For example, unsure whether the native-sounding phrase includes the determiner "a", a user might search for both quoted strings "English as Second Language" and "English as a Second Language". The counts obtained for each of these phrases on three different search engines are shown in Table 1. Note the correct version, "English as a Second Language", has a much higher number of web hits.

In order to determine whether this approach holds promise, we implemented a web-based system for ESL writing error proofing. This pilot study was intended to:

1. identify different types of ESL writing errors and how often they occur in ESL users' writing samples, so that the challenges and difficulties of ESL error proofing can be understood better;

2. explore the advantages and drawbacks of a web-

based approach, discover useful web data features, and identify which types of ESL errors can be reliably proofed using this technique.

We first catalog some major categories of ESL writing errors, then review related work. Section 3 describes our Web-based English Proofing System for ESL users (called **ESL-WEPS** later). Section 4 presents experimental results. Section 5 concludes.

1.1 ESL Writing Errors

In order to get ESL writing samples, we employed a third party to identify large volumes of ESL web pages (mostly from Japanese, Korean and Chinese ESL users' blogs), and cull 1K non-native sentences. A native speaker then rewrote these ESL sentences – when possible – to produce a native-sounding version. 353 (34.9%) of the original 1012 ESL sentences were labeled "native-like", another 347 (34.3%) were rewritten, and the remaining 312 (30.8%) were classified as simply unintelligible.

Table 2 shows some examples from the corpus illustrating some typical types of ESL writing errors involving: (1) Verb-Noun Collocations (VNC) and (4) Adjective-Noun Collocations (ANC); (2) incorrect use of the transitive verb "attend"; (3) determiner (article) usage problems; and (5) more complex lexical and style problems. We analyzed all the pre- and post-edited ESL samples and found 441 ESL errors: about 20% are determiner usage problems(missing/extra/misused); 15% are VNC errors, 1% are ANC errors; others represent complex syntactic, lexical or style problems. Multiple errors can co-occur in one sentence. These show that real-world ESL error proofing is very challenging.

Our findings are consistent with previous research results on ESL writing errors in two respects:

1. ESL users have significantly more problems with determiner usage than native speakers because the use and omission of definite and indefinite articles varies across different languages (Schneider and McCoy, 1998)(Lonsdale and Strong-Krause, 2003).

2. Collocation errors are common among ESL users, and collocational knowledge contributes to the difference between native speakers and ESL learners (Shei and Pain, 2000): in CLEC, a real-world Chinese English Learner Corpus (Gui and Yang, 2003), about 30% of ESL writing errors involve different types of collocation errors.

In the remainder of the paper, we focus on proofing determiner usage and VNC errors.

2 Related Work

Researchers have recently proposed some successful learning-based approaches for the determiner selection task (Minnen et al., 2000), but most of this work has aimed only at helping native English users correct typographical errors. Gamon *et al.*(2008) recently addressed the challenging task of proofing writing errors for ESL users: they propose combining contextual speller techniques and language modeling for proofing several types of ESL errors, and demonstrate some promising results. In a departure from this work, our system directly uses web data for the ESL error proofing task.

There is a small body of previous work on the use of online systems aimed at helping ESL learners correct collocation errors. In Shei and Pain's system (2000), for instance, the *British National Corpus (BNC)* is used to extract English collocations, and an ESL learner writing corpus is then used to build a collocation Error Library. In Jian *et al.*'s system (2004), the *BNC* is also used as a source of collocations, with collocation instances and translation counterparts from the bilingual corpus identified and shown to ESL users. In contrast to this earlier work, our system uses the web as a corpus, with string frequency counts from a search engine index used to indicate whether a particular collocation is being used correctly.

3 Web-based English Proofing System for ESL Users (ESL-WEPS)

The architecture of **ESL-WEPS**, which consists of four main components, is shown in Fig.1.

Parse ESL Sentence and Identify Check Points ESL-WEPS first tags and chunks (Sang and Buckholz, 2000) the input ESL sentence[1], and identifies the elements of the structures in the sentence to be checked according to certain heuristics: when

[1] One in-house HMM chunker trained on English Penn Treebank was used.

ID	Pre-editing version	Post-editing version
1	Which team can ***take*** the champion?	Which team will ***win*** the championship?
2	I ***attend to*** Pyoung Taek University.	I ***attend*** Pyoung Taek University.
3	***I'm a Japanese*** and studying Info and Computer Science at Keio University.	***I'm Japanese*** and studying Info Computer Science at Keio University.
4	Her works are ***kinda*** erotic but they will never arouse any obscene, ***devil thoughts*** which might destroy the soul of the designer.	Her works are ***kind of*** erotic, but they will never arouse any obscene, ***evil thoughts*** which might destroy the soul of the designer.
5	I think it is so beautiful to ***go the way of theology and very attractive too, especially in the area of Christianity***.	I think it is so beautiful to ***get into theology, especially Christianity, which attracts me***.

Table 2: Some pre- and post-editing ESL writing samples, Bold Italic characters show where the ESL errors are and how they are corrected/rewritten by native English speaker.

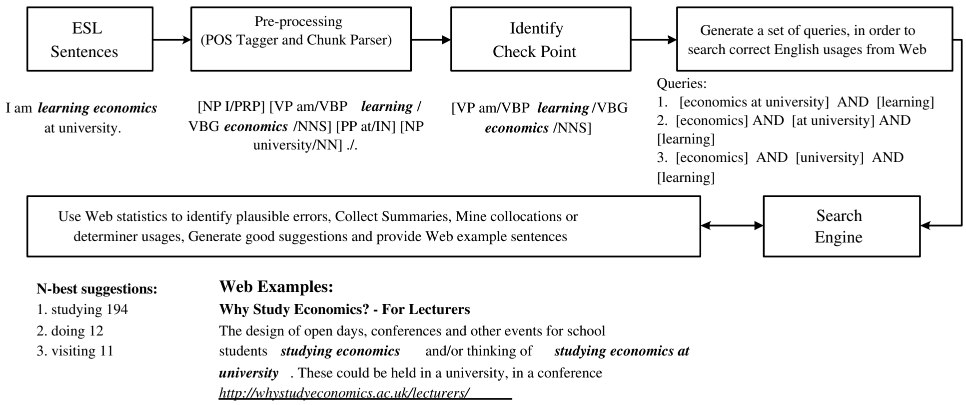


Figure 1: System Architecture

checking VNC errors, the system searches for a structure of the form (VP)(NP) or (VP)(PP)(NP) in the chunked sentence; when checking determiner usage, the system searches for (NP). Table 3 shows some examples. For efficiency and effectiveness, the user can specify that only one specific error type be critiqued; otherwise it will check both error types: first determiner usage, then collocations.

Generate Queries In order to find appropriate web examples, ESL-WEPS generates at each check point a set of queries. These queries involve three different granularity levels, according to sentence's syntax structure:

1. **Reduced Sentence Level**. In order to use more contextual information, our system preferentially generates a maximal-length query hereafter called *S-Queries*, by using the original sentence. For the check point chunk, the verb/adj. to be checked is found and extracted based on POS tags; other chunks are simply concatenated and used to formulate the query. For example, for the first example in Table 3, the S-Query is ['I have' AND 'this person for years' AND 'recognized'].

2. **Chunk Level**. The system segments each ESL sentence according to chunk tags and utilizes chunk pairs to generate a query, hereafter referred to as a *C-Query*, e.g. the C-Query for the second example in Table 3 is ['I' AND 'went' AND 'to climb' AND 'a tall mountain' AND 'last week']

3. **Word Level**. The system generates queries by using keywords from the original string, in the processing eliminating stopwords used in typical IR engines, hereafter referred to as a *W-Query*, e.g. W-Query for the first example in Table 3 is ['I' AND 'have' AND 'person' AND 'years' AND 'recognized']

As queries get longer, web search engines tend to return fewer and fewer results. Therefore, ESL-WEPS first segments the original ESL sentence by using punctuation characters like commas and semicolons, then generates a query from only the part which contains the given check point. When checking determiner usage, three different cases (a or an/the/none)

Parsed ESL sentence	Error Type	Check Points
(NP I/PRP) (VP have/VBP recognized/VBN) (NP this/DT person/NN) (PP for/IN) (NP years/NNS) ./.	VNC	recognized this person
(NP I/PRP) (VP went/VBD) (VP to/TO climb/VB) (NP a/DT tall/JJ mountain/NN) (NP last/JJ week/NN) ./.	ANC	tall mountain, last week
(NP I/PRP) (VP went/VBD) (PP to/TO) (NP coffee/NN) (NP shop/NN) (NP yesterday/NN) ./.	Determiner usage	coffee, shop, yesterday
(NP Someone/NN) (ADVP once/RB) (VP said/VBD) (SBAR that/IN) (ADVP when/WRB) (NP you/PRP) (VP meet/VBP) (NP a/DT right/JJ person/NN) (PP at/IN) (NP the/DT wrong/JJ time/NN),/, (NP it/PRP) (VP 's/VBZ) (NP a/DT pity/NN)./.	Determiner usage	meet a right person at the wrong time 's a pity

Table 3: Parsed ESL sentences and Check Points.

are considered for each check point. For instance, given the last example in Table 3, three C-Queries will be generated: [meet a right person],[meet the right person] and [meet right person]. Note that a term which has been POS-tagged as NNP (proper noun) will be skipped and not used for generating queries in order to obtain more web hits.

Retreive Web Statistics, Collect Snippets To collect enough web examples, three levels of query sets are submitted to the search engine in the following order: S-Query, C-Query, and finally W-Query. For each query, the web hits *df* returned by search engine is recorded, and the snippets from the top 1000 hits are collected. For efficiency reasons, we follow Dumais (2002)'s approach: the system relies only on snippets rather than full-text of pages returned for each hit; and does not rely on parsing or POS-tagging for this step. However, a lexicon is used in order to determine the possible parts-of-speech of a word as well as its morphological variants. For example, to find the correct VNC for a given noun 'tea' in the returned snippets, the verb **drank** in the same clause will be matched before 'tea'.

Identify Errors and Mine Correct Usages To detect determiner usage errors, both the web hit df_q and the length l_q of a given query q are utilized, since longer query phrases usually lead to fewer web hits. DFL_q, $DFLMAX$, q_{max} and R_q are defined as:

$$\begin{aligned}
&DFL_q = df_q \times l_q, for\ a\ given\ query\ q;\\
&DFLMAX = \max(DFL_q),\\
&q_{max} = \arg\max_q(DFL_q),\\
&q \in \{queries\ for\ a\ given\ check\ point\};\\
&R_q = DFL_q/DFLMAX, given\ query\ q\ and\ check\ point.
\end{aligned}$$

If $DFLMAX$ is less than a given threshold t_1, this check point will be skipped; otherwise the q_{max} indicates the best usage. We also calculate the relative ratio R_q for three usages (a or an/the/none). If R_q is larger than a threshold t_2 for a query q, the system will not report that usage as an error because it is sufficiently supported by web data.

For collocation check points, ESL-WEPS may interact twice with the search engine: first, it issues query sets to collect web examples and identify plausible collocation errors; then, if errors are detected, new query sets will be issued in the second step in order to mine correct collocations from new web examples. For example, for the first sentence in Table 3, the S-Query will be ['I have' AND 'this person for years' AND 'recognized']; the system analyzes returned snippets and identifies 'recognized' as a possible error. The system then issues a new S-Query ['I have' AND 'this person for years'], and finally mines the new set of snippets to discover that 'known' is the preferred lexical option. In contrast to proofing determiner usages errors, $mfreq$:

$$\begin{aligned}
&mfreq = frequency\ of\ matched\ collocational\ verb/adj.\\
&in\ the\ snippets\ for\ a\ given\ noun,
\end{aligned}$$

is utilized to both identify errors and suggest correct VNCs/ANCs. If $mfreq$ is larger than a threshold t_3, the system will conclude that the collocation is plausible and skip the suggestion step.

4 Experiments

In order to evaluate the proofing algorithm described above, we utilized the MSN search engine API and the ESL writing sample set described in Section 1.1 to evaluate the algorithm's performance on two tasks: determiner usage and VNC proofing. From a practical standpoint, we consider precision on the proofing task to be considerably more important than recall: false flags are annoying and highly visible to the user, while recall failures are much less problematic.

Given the complicated nature of the ESL error proofing task, about 60% of ESL sentences in our set that contained determiner errors also contained other types of ESL errors. As a result, we were forced to slightly revise the typical precision/recall measurement in order to evaluate performance. First,

Good Proofing Examples	
Error sentence 1	In my opinion, therefore, when we describe terrorism, its crucially important that we ***consider the degree of the influence*** (i.e., power) on the other countries.
proofing suggestion	consider the degree of influence
Error sentence 2	Someone once said that when you ***meet a right person at the wrong time***, it's a pity.
proofing suggestion	meet the right person at the wrong time
Plausible Useful Proofing Examples	
Error sentence 3	The most powerful place in Beijing, and ***in the whole China***.
native speaker suggestion	in the whole of China
system suggestion	in whole China
Error sentence 4	Me, I wanna keep in touch with old friends and wanna talk with anyone who ***has different thought***, etc.
native speaker suggestion	has different ideas
system suggestion	has a different thought

Table 4: ESL Determiner Usage Proofing by Native Speaker and ESL-WEPS.

Good Proofing Examples	
Error sentence 1	I had great time there and ***got many friends***.
proofing suggestion	made many friends
Error sentence 2	Which team can ***take the champion***?
proofing suggestion	win the champion
Plausible Useful Proofing Examples	
Error sentence 3	It may ***sounds fun*** if I say my firm resolution of this year is to get a girl friend.
native speaker suggestion	sound funny
system suggestion	make * fun or get * fun

Table 5: ESL VNC Proofing by Native Speaker and ESL-WEPS.

we considered three cases: (1) the system correctly identifies an error and proposes a suggestion that exactly matches the native speaker's rewrite; (2) the system correctly identifies an error but makes a suggestion that differs from the native speaker's edit; and (3) the system incorrectly identifies an error. In the first case, we consider the proofing *good*, in the second, *plausibly useful*, and in the third case it is simply *wrong*. Correspondingly, we introduce the categories *Good Precision (GP)*, *Plausibly Useful Precision (PUP)* and *Error Suggestion Rate (ESR)*, which were calculated by:

$$GP = \frac{\#\ of\ Good\ Proofings}{\#\ of\ System's\ Proofings};$$
$$PUP = \frac{\#\ of\ Plausibly\ Useful\ Proofings}{\#\ of\ System's\ Proofings};$$
$$ESR = \frac{\#\ of\ Wrong\ Proofings}{\#\ of\ System's\ Proofings};$$
$$GP + PUP + ESR = 1$$

Furthermore, assuming that there are overall N_A errors for a given type A of ESL error , the typical *recall* and *false alarm* were calculated by:

$$recall = \frac{\#\ of\ Good\ Proofings}{N_A};$$
$$false\ alarm = \frac{\#\ of\ Wrong\ Proofings}{\#\ of\ Check\ points\ for\ ESL\ error\ A}$$

Table 4 and Table 5 show examples of ***Good*** or ***Plausibly Useful*** proofing for determiner usage and collocation errors, respectively. It can be seen the system makes plausibly useful proofing suggestions because some errors types are out of current system's checking range.

The system achieved very promising performance despite the fact that many of the test sentences contained other, complex ESL errors: using appropriate system parameters, ESL-WEPS showed recall 40.7% on determiner usage errors, with 62.5% of these proofing suggestions exactly matching the rewrites provided by native speakers. Crucially, the false flag rate was only 2%. Note that a random-guessing baseline was about 5% precision, 7% recall, but more than 80% false flag rate.

For collocation errors, we focused on the most common VNC proofing task. $mfreq$ and threshold t_3 described in Section 3 are used to control false alarm, GP and recall. A smaller t_3 can reduce recall, but can increase GP. Table 7 shows how performance changes with different settings for t_3, and Fig. 2(b) plots the GP/recall curve. Results are not very good: as recall increases, GP decreases too quickly, so that at 30.7% recall, precision is only 37.3%. We attribute this to the fact that most search engines only return the top 1000 web snippets for each query and our current system relies on this limited number of snippets to generate and rank candidates.

Recall	16.3%	30.2%	**40.7%**	44.2%	47.7%	50.0%
GP	73.7%	70.3%	**62.5%**	56.7%	53.3%	52.4%
PUP	15.8%	16.2%	**25.0%**	29.9%	29.9%	29.3%
false alarm	0.4%	1.4%	**2.0%**	2.6%	3.7%	4.3%

Table 6: Proofing performance of determiner usage changes when setting different system parameters.

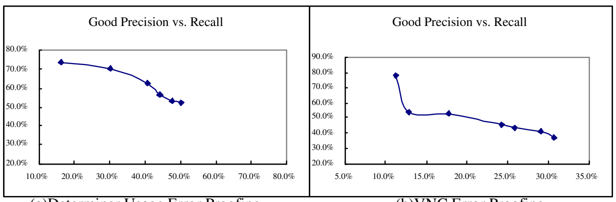
Recall	11.3%	12.9%	17.8%	25.8%	29.0%	30.7%
GP	77.8%	53.3%	52.4%	43.2%	40.9%	37.3%
PUP	11.11%	33.33%	33.33%	45.10%	48.65%	50.00%
false alarm	0.28%	0.57%	0.85%	0.85%	1.13%	2.55%

Table 7: VNC Proofing performance changes when setting different system parameters.

5 Conclusion

This paper introduced an approach to the challenging real-world ESL writing error proofing task that uses the index of a web search engine for corpus statistics. We validated ESL-WEPS on a web-crawled ESL writing corpus and compared the system's proofing suggestions to those produced by native English speakers. Promising performance was achieved for proofing determiner errors, but less good results for VNC proofing, possibly because the current system uses web snippets to rank and generate collocation candidates. We are currently investigating a modified strategy that exploits high quality local collocation/synonym lists to limit the number of proposed Verb/Adj. candidates.

We are also collecting more ESL data to validate our system and are extending our system to more ESL error types. Recent experiments on new data showed that ESL-WEPS can also effectively proof incorrect choices of prepositions. Later research will compare the web-based approach to conventional corpus-based approaches like Gamon *et al.* (2008), and explore their combination to address complex ESL errors.



(a)Determiner Usage Error Proofing (b)VNC Error Proofing

Figure 2: GP/recall curves. X and Y axis denotes GP and Recall respectively.

Acknowledgement The authors have benefited extensively from discussions with Michael Gamon and Chris Brockett. We also thank the Butler Hill Group for collecting the ESL examples.

References

C. Brockett, W. B. Dolan, and M. Gamon. 2006. Correcting ESL errors using phrasal smt techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 249–256, Sydney, Australia.

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. 2002. Web question answering: is more always better? In *Proceedings of the 25th Annual International ACM SIGIR*, pages 291–298, Tampere, Finland.

M. Gamon, J.F. Gao, C. Brockett, A. Klementiev, W.B. Dolan, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP 2008*, Hyderabad, India, January.

S. Gui and H. Yang, 2003. *Zhongguo Xuexizhe Yingyu Yuliaoku. (Chinese Learner English Corpus)*. Shanghai Waiyu Jiaoyu Chubanshe, Shanghai. (In Chinese).

Jia-Yan Jian, Yu-Chia Chang, and Jason S. Chang. 2004. TANGO: bilingual collocational concordancer. In *Proceedings of the ACL 2004*, pages 19–23, Barcelona, Spain.

D. Lonsdale and D. Strong-Krause. 2003. Automated rating of ESL essays. In *Proceedings of the NAACL-HLT 03 workshop on Building educational applications using natural language processing*, pages 61–67, Edmonton, Canada.

G. Minnen, F. Bond, and A. Copestake. 2000. Memory-based learning for article generation. In *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, pages 43–48.

E. Tjong Kim Sang and S. Buckholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal.

D. Schneider and K. F. McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proceedings of the 17th international conference on Computational linguistics*, pages 1198–1204, Montreal, Quebec, Canada.

C.-C. Shei and H. Pain. 2000. An esl writer's collocational aid. *Computer Assisted Language Learning*, 13(2):167–182.

Analysis of Intention in Dialogues Using Category Trees and Its Application to Advertisement Recommendation

Hung-Chi Huang **Ming-Shun Lin** **Hsin-Hsi Chen**
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
{hchuang, mslin}@nlg.csie.ntu.edu.tw; hhchen@ntu.edu.tw

Abstract

We propose an intention analysis system for instant messaging applications. The system adopts Yahoo! directory as category trees, and classifies each dialogue into one of the categories of the directory. Two weighting schemes in information retrieval, i.e., *tf* and *tf-idf*, are considered in our experiments. In addition, we also expand Yahoo! directory with the accompanying HTML files and explore different features such as nouns, verbs, hypernym, hyponym, *etc*. Experiments show that category trees expanded with snippets together with noun features under *tf* scheme achieves a best F-score, 0.86, when only 37.46% of utterances are processed on the average. This methodology is employed to recommend advertisements relevant to the dialogue.

1 Introduction

Instant messaging applications such as Google Talk, Microsoft MSN Messenger, Yahoo Messenger, QQ, and Skype are very popular. In the blooming instant messaging markets, sponsor links and advertisements support the free service. Figure 1 shows an example of sponsor links in instant message applications. They are usually randomly proposed and may be irrelevant to the utterance. Thus, they may not attract users' attentions and have no effects on advertisements. This paper deals with the analysis of intention in the dialogues and the recommendation of relevant sponsor links in an ongoing conversation.

In the related works, Fain and Pedersen (2006) survey sponsored search, suggesting the importance of matching advertising content to user intentions. How to match advertiser content to user queries is an important issue. Yih et al. (2006) aimed at extracting advertisement keywords from the intention on the web pages. However, these works did not address the issues in dialogues.



Figure 1. A Sponsor Link in an IM Application

In conventional dialogue management, how to extract semantic concepts, identify the speech act, and formulate the dialogue state transitions are important tasks. The domain shift is a challenging problem (Lin and Chen, 2004). In instant message applications, more challenging issues have to be tackled. Firstly, the discussing topics of dialogues are diverse. Secondly, the conversation may be quite short, so that the system should be responsive instantly when detecting the intention. Thirdly, the utterance itself can be purely free-style and far beyond the formal grammar. That is, self-defined or symbolic languages may be used in the dialogues. The following shows some example utterances.

James: dud, i c ur foto on Kelly's door~ ^^||

Antony: Orz….kill me pls. ><

An intention detecting system has to extract words from incomplete sentences in dialogues. Fourthly, the system should consider up-to-date terms, instead of just looking up conventional dictionaries.

Capturing the intention in a dialogue and recommending the advertisements before its ending are the goal of this approach. This paper is organized as follows. Section 2 shows an overview of the system architecture. Section 3 discusses the category trees and the weighting functions for identifying the intention. Section 4 presents the experimental results comparing with different uses of the category trees and word features. Section 5 concludes and remarks.

2 System Overview

Fain and Pedersen (2006) outlined six basic elements for sponsored search. They are shown as follows:

(1) advertiser-provided content,
(2) advertiser-provided bids,
(3) ensuring that advertiser content is relevant to the target keyword,
(4) matching advertiser content to user queries,
(5) displaying advertiser content in some rank order,
(6) gathering data, metering clicks and charging advertisers.

In instant messaging applications, a dialogue is composed of several utterances issuing by at least two users. They are different from sponsored search in that advertiser content is matched to user utterances instead of user queries. While reading users' conversation, an intention detecting system recommends suitable advertiser information at a suitable time. The time of the recommendation and the effect of advertisement have a strong relationship. The earlier the correct recommendation is, the larger the effect is.

However, time and accuracy are trade-off. At the earlier stages of a dialogue, the system may have deficient information to predict suitable advertisement. Thus, a false advertisement may be proposed. On the other hand, the system may have enough information at the later stages. However, users may complete their talk at any time in this case, so the advertisement effect may be lowered.

Figure 2 shows architecture of our system. In each round of the conversation, we retrieve an utterance from a given instant message application. Then, we parse the utterance and try to predict intention of the dialogue based on current and previous utterances, and consult the advertisement databases that provide sponsor links accordingly. If the information in the utterances is enough for prediction, then several candidates are proposed. Finally, based on predefined criteria, the best candidate is selected and proposed to the IM application as the sponsor link in Figure 1.

In the following sections, we will explore when to make sure the intention of a dialogue with confidence and to propose suitable recommendations. In addition, we will also discuss what word features (called *cue words* hereafter) in the utterances are useful for the intention determination. We assume sponsor links or advertisements are adjunct on the given category trees.

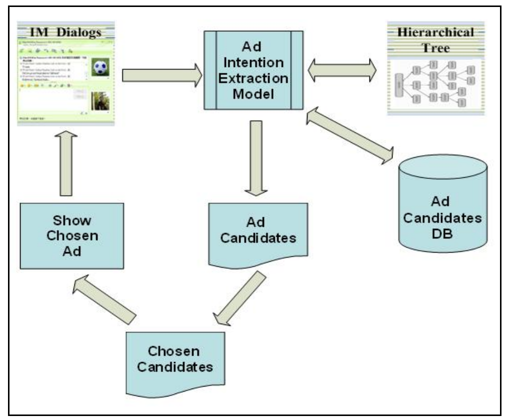


Figure 2. System Architecture

3 Categorization of Dialogues

3.1 Web Directory Used for Categorization

We employ Yahoo! directory[1] to assign a dialogue or part of a dialogue in category representing its intention. Every word in dialogues is classified by the directory. For example, by searching the term *BMW*, we could retrieve the category path:

>Business and Economy>… Makers>Vehicles

Each category contains subcategories, which include some subsidiary categories. Therefore, we could take the directory as a hierarchical tree for searching the intention. Moreover, each node of the tree has attributes from the node itself and its ancestors. Our idea is to summarize all intentions from words in a dialog, and then conclude the intention accordingly.

The nodes sometimes are overlapped, that is, one node could be found in more than one path. For example, the car maker *BMW* has at least two other nodes:

[1] http://dir.yahoo.com

>Regional>Countries>Germany>Business and Economy>…>Dealers
>Recreation>Automotive>…Clubs and Organizations>BMW Car Club of America

The categories of *BMW* include *Business and Economy*, *Regional*, and *Recreation*. This demonstrates the nature of the word ambiguity, and is challenging when the system identifies the intention embedded in the dialogs.

The downloaded Yahoo! directory brings up HTML documents with three basic elements, including titles, links and snippet as shown in Figure 3. The following takes the three elements from a popular site as an example.

Title: *The White House*
Link: *www.WhiteHouse.gov*
Snippet: *Features statements and press releases by President George W. Bush as well…*



Figure 3. Sample HTML in Yahoo! Directory Tree

We will explore different ways to use the three elements during intention identification. Table 1 shows different models and total nodes. YahooO and YahooX are two extreme cases. The former employs the original category tree, while the latter expands the category tree with titles, links and snippets. Thus, the former contains 7,839 nodes and the latter 78,519 nodes.

Tree	Scenario	Total
YahooO	Original Yahoo! directory	7839
YahooX	Yahoo! directory expanded with web title, link, and snippet	78519
YahooS	Yahoo! directory expanded with web snippet	32476
YahooT	Yahoo! directory expanded with web title	39207
YahooL	Yahoo! directory expanded with web link	19690

Table 1. Tree Expansion Scenarios

Tree	Example
YahooT	..\Computers_and_Internet\~W3C HTML Validation Service ..\News_and_Media\~YouTube ..\Science\~Scientific American ..\Social_Science\~Amnesty International ..\Society_and_Culture\~MySpace ..Recreation\Gambling\~Mahjong
YahooS	..\Arts\Artists\Masters\~ Yann Arthus-Bertrand is a photojournalist specializing in adventure, sports, and nature photography. His best know works are aerial shots taken from a hot air balloon. ..\Business_and_Economy\Shopping_and_Services\ ~AltaVista provides web and newsgroup search engines, as well as paid submission services ..\Computers_and_Internet\Desktop_Publishing\~ Converts Mac Word files to Quark Xpress
YahooL	..\Computers_and_Internet\~www.java.sun.com ..\Health\~www.webmd.com ..\Education\Programs\~www.sea.edu ..\Entertainment\~www.pandora.com ..\News_and_Media\~www.nytimes.com ..\News_and_Media\~www.youtube.com

Table 2. Examples of Expanded Nodes

Table 2 lists some examples to demonstrate the category tree expansion. Some words inside the three elements rarely appear in dictionaries or encyclopedias. Thus, we can summarize these trees and build a new dictionary with definitions. For example, we could find the hottest web sites ***YouTube*** and ***MySpace***, and even the most popular Chinese gamble game, ***Mahjong***.

3.2 Scoring Functions for Categorization

Given a fragment *F* of a dialogue, which is composed of utterances reading up to now, Formula 1 determines the intention I_{INT} of *F* by counting total scores of *cue words w* in *F* contributing to *I*.

$$I_{INT} = \arg\max \sum_{w \in F} tf(w) \times b(w, I) \quad (1)$$

where $tf(w)$ is term frequency of w in F, and $b(w,I)$ is 1 when w is in the paths corresponding to the intention I_{INT}; $b(w,I)$ is 0 otherwise.

Formula 2 considers the discriminating capability of each cue word. It is similar to *tf-idf* scheme in information retrieval.

$$I_{INT} = \underset{I}{\operatorname{argmax}} \sum_{w \in F} tf(w) \times \log \frac{N}{df(w)} \times b(w, I) \quad (2)$$

where N is total number of intentions, and $df(w)$ is total intentions in which w appears.

3.3 Features of Cue Words

The features of possible cue words including nouns, verbs, stop-words, word length, hypernym, hyponym, and synonym are summarized in Table 3 with explanation and examples.

Cue word	Explanation	Source	Sample
Noun	Words defined as nouns	Dictionary	Buddha, Earth
Verb	Words defined as verbs	Dictionary	Go, get, replace
Stopword	Words used often but not meaningful	Dictionary	Of, on, off, alongside
Length	Total number of letter in a word	Calculation	Length of "on" is two
Hypernym	A superclass of a word	WordNet	Dog is a hypernym of puppy
Hyponym	A subclass of a word	WordNet	Puppy is a hyponym of dog
Synonym	Same meaning words	WordNet	Feline is a synonym of cat

Table 3. Cue Words Explored

Nouns and verbs form skeletons of concepts are important cues for similarity measures (Chen et al., 2003), so that they are considered as features in our model. Word length is used to filter out some unnecessary words because the shorter the word is, the less meaningful the word might be. Here we postulate that instant messaging users are not willing to type long terms if unnecessary.

In this paper, we regard words in an utterance of dialogues as query terms. Rosie et al. (2006) showed that query substitution may be helpful to retrieve more meaningful results. Here, we use hypernym, hyponym and synonym specified in WordNet (Fellbaum, 1998) to expand the original utterance.

3.4 Candidate Recommendation

The proposed model also provides the ability to show the related advertisements after intention is confirmed. As discussed, for each of node in the category tree, there is an accompanying HTML file to show some related web sites and even sponsors. Therefore, we can also use the category tree to put sponsor links into the HTML files, and just fetch the sponsor links from the HTML file on the node to the customers.

The algorithm to select the suitable candidates could be shortly described as the Longest Path First. Once we select the category of the intention, the nodes appearing in the chosen category will then be collected into a set. We will check the longest path and provide the sponsor links from the node.

4 Experimental Results

4.1 Performance of Different Models

To prepare the experimental materials, we collected 50 real dialogs from end-users, and asked annotators to tag the 50 dialogs with 14 given Yahoo! directory categories shown in Table 4. Average number of sentences is 12.38 and average number of words is 56.04 in each dialog. We compare the system output with the answer keys, and compute precision, recall, and F-score for each method.

Category	Abbreviation
Arts & Humanities	A
Business & Economy	B
Computers & Internet	C
Education	D
Entertainment	E
Government	F
Health	G
News & Media	H
Recreation & Sports	I
Reference	J
Regional	K
Science	L
Social Science	M
Society & Culture	N
na	X

Table 4. Category Abbreviation

Table 5 shows the performance of using Formula 1 (i.e., *tf* scheme). This model is a combination of a scenario shown in Table 1 and features shown in Table 3. For example, the YahooS-noun matches cue words of POS *noun* from utterances to the category tree expanded with *snippets*. WL denotes word length. Only cue words of length ≥ WL is considered. C denotes the number of dialogues correctly analyzed. NA denotes the number of undecidable dialogues. P, R and F denote precision, recall and F-score.

Table 5 shows that YahooS with noun features achieves a best performance. Noun feature works impressively well with the orders, YahooS, YahooT, YahooX, and YahooL. That meets our expectation because the information from snippets is well enough and does not bring in noise as the YahooX. YahooT, however, has good but insufficient information, while YahooL is only suitable for dialogs directly related to links.

Moreover, the experimental results show that verb is not a good feature no matter whether the category tree is expanded or not. Although some verbs can explicitly point out the intention of dialogues, such as *buy*, *sell*, *purchase*, *etc*, the lack of verbs in Yahoo! directory makes the verb features less useful in the experiments. Table 6 shows the performance of using Formula 2 (i.e., *tf-idf* scheme). The original category tree with hyponym achieves the best performance, i.e., 56.56%. However, it cannot compete with most of models with *tf* scheme.

Model	WL	C	NA	P	R	F
YahooS noun	0	43	0	86	86	86.00
YahooT noun	3	38	0	76	76	76.00
YahooX noun	3	37	0	74	74	74.00
YahooL noun	3	37	0	74	74	74.00
YahooS no-stopword	1	36	0	72	72	72.00
YahooO noun	3	34	1	69	68	68.69
YahooS hypernym	4	34	0	68	68	68.00
YahooS	4	33	0	66	66	66.00
YahooT	4	33	0	66	66	66.00
YahooX no-stopword	4	33	0	66	66	66.00
YahooT no-stopword	4	33	0	66	66	66.00
YahooX synonym	4	33	0	66	66	66.00
YahooX	4	32	0	64	64	64.00
YahooO noun verb	0	31	1	63	62	62.63
YahooO	3	31	1	63	62	62.63
YahooS hyponym	4	31	0	62	62	62.00
YahooL	4	31	0	62	62	62.00
YahooL no-stopword	4	30	0	60	60	60.00
YahooT synonym	3	29	0	58	58	58.00
YahooX hyponym	4	29	0	58	58	58.00
YahooL synonym	4	29	0	58	58	58.00
YahooT hyponym	4	28	0	56	56	56.00
YahooO hyponym	3	27	0	54	54	54.00
YahooL hyponym	4	26	0	52	52	52.00
YahooO hypernym	3	25	0	50	50	50.00
YahooT hypernym	5	23	0	46	46	46.00
YahooS verb hyponym	1	17	1	35	34	34.34
YahooS verb	1	15	1	31	30	30.30
YahooO verb	4	2	42	25	4	6.90

Table 5. Performance of Models with *tf* Scheme

Model	WL	C	NA	P	R	F
YahooO hyponym	3	28	1	57	56	56.56
YahooO no-stopword	3	28	1	57	56	56.56
YahooO	3	27	1	55	54	54.54
YahooO hypernym	3	26	1	53	52	52.52
YahooO noun	3	25	1	51	50	50.50
YahooL noun	4	18	0	36	36	36.00
YahooT no-stopword	3	17	0	34	34	34.00
YahooL	4	17	0	34	34	34.00
YahooT noun	3	17	0	34	34	34.00
YahooT noun, hyponym, hypernym	3	16	0	32	32	32.00
YahooT	3	15	0	30	30	30.00
YahooX noun	2	11	0	22	22	22.00
YahooS noun	3	10	0	20	20	20.00
YahooS no-stopword	4	9	0	18	18	18.00
YahooO verb	3	6	32	33	12	17.65
YahooS verb	4	7	18	22	14	17.07
YahooX	1	8	0	16	16	16.00
YahooX no-stopword	1	8	0	16	16	16.00
YahooS	4	7	0	14	14	14.00

Table 6. Performance of Models with *tf-idf* Scheme

4.2 Hit Speed

Besides precision, recall and F-score, we are also interested if the system captures the intention of the dialogue at better timing. We define one more metric called *hit speed* in Formula (3). It represents how fast the sponsor links could be correctly suggested during the progress of conversations. For each utterance in a dialogue, we mark either *X* or a predicted category. Here *X* denotes *undecidable*.

Assume we have a dialogue of 7 utterances and consider the following scenario. At first, our system could not propose any candidates in the first two utterances. Then, it decides the third and the fourth utterances are talking about *Business and Economy*. Finally, it determines the intention of the dialogue is *Computer and Internet* after reading the next three utterances. In this example, we get an *answer string*, *XXBBCCC*, based on the notations shown in Table 4. If the intention annotated by human is *Computer and Internet*, then the system starts proposing a correct intention from the 5th utterance. In other words, the information in the first 4 utterances is not sufficient to make any decision or make wrong decision.

Let *CPL* be the length of correct postfix of an answer string, e.g., 3, and *N* be total utterances in a dialogue, e.g., 7. *HitSpeed* is defined as follows.

$$HitSpeed = \frac{CPL}{N} \quad (3)$$

In this case, the *hit speed* of intention identification is 3/7. Intuitively, our goal is to get the hit speed as high as possible. The sooner we get the correct intention, the better the recommendation effect is.

The average hit speed is defined by Formulas (4) and (5). The former considers only the correct dialogues, and the latter considers all the dialogues. Let *M* and *N* denote total dialogues and total correct dialogues, respectively.

$$AvgHitSpeed = \frac{\sum_{i=1}^{M} HitSpeed_i}{N} \quad (4)$$

$$AvgHitSpeed = \frac{\sum_{i=1}^{M} HitSpeed_i}{M} \quad (5)$$

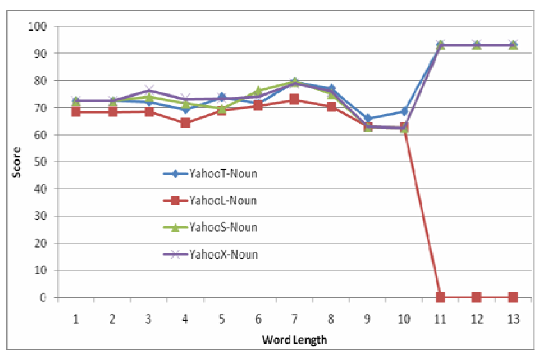


Figure 4. Average Hit Speed by Formula (4)

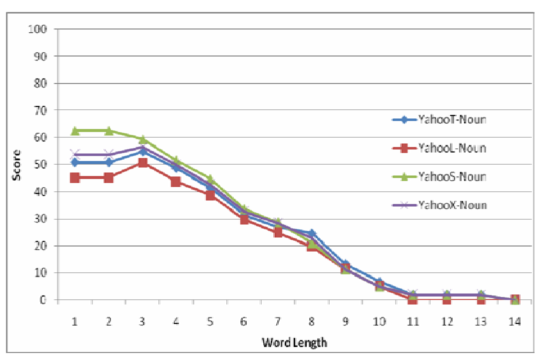


Figure 5. Average Hit Speed by Formula (5)

Figures 4 and 5 demonstrate average hit speeds computed by Formulas (4) and (5), respectively. Here four leading models shown in Table 5 are adopted and nouns are regarded as cue words. Figure 4 shows that the average hit speed in correctly answered dialogues is around 70%. It means these models can correctly answer the intention when a dialogue still has 70% to go in the set of correctly answered dialogs.

Figure 5 considers all the dialogues no matter whether their intentions are identified correctly or not. We can still capture the intention with the hit speed 62.54% for the best model, i.e., YahooS-noun.

5 Concluding Remarks

This paper captures intention in dialogues of instant messaging applications. A web directory such as Yahoo! directory is considered as a category tree. Two schemes, revised *tf* and *tf-idf*, are employed to classify the utterances in dialogues. The experiments show that the *tf* scheme using the category tree expanded with snippets together with noun features achieves the best F-score, 0.86. The hit speed evaluation tells us the system can start making good decision when near only 37.46% of total utterances are processed. In other words, the recommended advertisements can be placed to attract users' attentions in the rest 62.54% of total utterances.

Though the best model in the experiments is to use nouns as features, we note that another important language feature, verbs, is not helpful due to the characteristic of the category tree we adopted, that is, the absence of verbs in Yahoo! directory. If some other data sources can provide the cue information, verbs may be taken as useful features to boost the performance.

In this paper, only one intention is assigned to the utterances. However, there may be many participants involving in a conversation, and the topics they are talking about in a dialogue may be more than one. For example, two couples are discussing a trip schedule together. After the topic is finished, they may continue the conversation for selection of hotels and buying funds separately in the same instant messaging dialogue. In this case, our system only decides the intention is *Recreation*, but not including *Business & Economy*.

Long time delay of response is another interesting topic for instant messaging dialogues. Sometimes one participant could send a message, but have to wait for minutes or even hours to get response. Because the receiver might be absent, busy or just off-line, the system should be capable of waiting such a long time delay before a complete dialogue is finished in practical applications.

Opinion mining is also important to the proposed model. For example, dialogue participants may talk about buying digital cameras, and one of them has negative opinions on some products. In such a case, an intelligent recommendation system should not promote such products. Once opinion extraction is introduced to intention analysis systems, customers can get not only the conversation-related, but also personally preferred sponsor links.

Acknowledgments

Research of this paper was partially supported by Excellent Research Projects of National Taiwan University, under the contract 95R0062-AE00-02.

References

H.H. Chen, J.J. Kuo, S.J. Huang, C.J. Lin and H.C. Wung. 2003. A Summarization System for Chinese News from Multiple Sources. *Journal of American Society for Information Science and Technology*, 54(13), pp. 1224-1236.

D. C. Fain and J. O. Pedersen. 2006. Sponsored Search: A Brief History. *Bulletin of the American Society for Information Science and Technology*, January.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating Query Substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, 2006, pp. 387-396.

K.K. Lin and H.H. Chen. 2004. Extracting Domain Knowledge for Dialogue Model Adaptation. In *Proceedings of 5th International Conference on Intelligent Text Processing and Computational Linguistics*, Lecture Notes in Computer Science, LNCS 2945, Springer-Verlag, pp. 70-78.

W. Yih, J. Goodman, and V. R. Carvalho. 2006. Finding Advertising Keywords on Web Pages. In *Proceedings of the 15th International Conference on World Wide Web*, pp. 213-222.

Term Extraction Through Unithood And Termhood Unification

Thuy VU, Ai Ti AW, Min ZHANG
Department of Language Technology, Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
{tvu, aaiti, mzhang}@i2r.a-star.edu.sg

Abstract

Term Extraction (TE) is an important component of many NLP applications. In general, terms are extracted for a given text collection based on global context and frequency analysis on words/phrases association. These extracted terms represent effectively the text content of the collection for knowledge elicitation tasks. However, they fail to dictate the local contextual information for each document effectively. In this paper, we refine the state-of-the-art *C/NC-Value* term weighting method by considering both *termhood* and *unithood* measures, and use the former extracted terms to direct the local term extraction for each document. We performed the experiments on Straits Times year 2006 corpus and evaluated our performance using Wikipedia termbank. The experiments showed that our model outperforms *C/NC-Value* method for global term extraction by 24.4% based on term ranking. The precision for local term extraction improves by 12% when compared to pure linguistic based extraction method.

1 Introduction

Terminology Extraction (TE) is a subtask of information extraction. The goal of TE is to automatically extract relevant terms from a given corpus. These extracted terms are used in a variety of NLP tasks such as information retrieval, text mining, document summarization etc. In our application scenario, we are interested in terms whose constituent words have strong collocation relations and can be translated to another language in stable single word or multi-word translation equivalents. Thus, we define “term” as a word/phrase that carries a special meaning.

A general TE consists of two steps. The first step makes use of various degrees of linguistic filtering (e.g., part-of-speech tagging, phrase chunking etc.), through which candidates of various linguistic patterns are identified (e.g. noun-noun, adjective-noun-noun combinations etc.). The second step involves the use of frequency- or statistical-based evidence measures to compute weights indicating to what degree a candidate qualifies as a terminological unit. There are many methods in literature trying to improve this second step. Some of them borrowed the metrics from Information Retrieval to evaluate how important a term is within a document or a corpus. Those metrics are Term Frequency/Inverse Document Frequency (TF/IDF), Mutual Information, T-Score, Cosine, and Information Gain. There are also other works (Nakagawa and Mori, 2002; Frantzi and Ananiadou, 1998) that introduced better method to weigh the term candidates.

Currently, the C/NC method (Frantzi and Ananiadou, 1998) is widely considered as the state-of-the-art model for TE. Although this method was first applied on English, it also performed well on other languages such as Japanese (Hideki Mima and Sophia Ananiadou, 2001), Slovene (Špela Vintar, 2004), and other domains such as medical corpus (Frantzi and Ananiadou, 1998), and computer science (E. Milios et al, 2003).

In terminology research, a term is evaluated using two types of feature: termhood[1] and unithood

[1] Termhood refers to a degree of linguistic unit. It considers a term as a linguistic unit representative for the document content.

[2](Kyo Kageura, 1996). In C/NC method, the features used to compute the term weight are based on termhood only. In this paper, we introduce a unithood feature, T-Score, to the C/NC method. Experiment results show that by incorporating T-Score into C/NC to derive a new weight, $NTCValue$, it gives a better ranking of the global terms and outperforms *C/NC* method by 24.4%.

On the other hand, C/NC method extracts term candidates using linguistic patterns and derives their weights based on distribution of terms over all documents. The extracted terms thus represent global content of the corpus, and do not represent well the contextual information for each individual document. So, we propose a method to enrich the local terms through a Term Re-Extraction Model (TREM). Experiment results show that the precision for local TE has been improved significantly, by 12% when compared to pure linguistic based extraction method.

In the following sections, we introduce the state-of-the-art method, the C/NC Value method. We then introduce our proposed methods, the *NTCValue* method on section 3, the Term Re-Extraction Model (TREM) on section 4 followed by the experiment results and conclusion.

2 The C/NC value Method

C/NC method uses a combination of linguistic and statistical information to evaluate the weight of a term. This method has two steps: candidate extraction and term weighting by C/NC value.

2.1 Term Candidate Extraction

This method uses 3 linguistic patterns to extract the term candidates:

- → (Noun+Noun);
- → (Adj|Noun)+Noun;
- → (Adj|Noun)+|((Adj|Noun)*(NounPrep)?)(Adj|Noun)*)Noun.

The term candidates are passed to the second step.

2.2 Term Weighting

2.2.1 CValue

$CValue$ is calculated based on the frequency of term and its subterms.

$$CValue(a) = \log_2 |a| \cdot \left(f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b) \right)$$

Where, $f(a)$ is the frequency of term a with $|a|$ words, T_a is the set of extracted candidate terms that contain a and $P(T_a)$ is the total number of longer candidate terms that contain a. The formula $\frac{1}{P(T_a)} \sum_{b \in T_a} f(b)$ will have value 0 when T_a is empty.

2.2.2 NC Value

$NCValue$ combines the context information of a term together with the CValue. The weight of a context word[3] b is defined by the number of terms $t(b)$ in which it appears over the total number of terms considered, n. C_a is the set of distinct context words and $f_a(b)$ is the frequency of b as context word of a.

$$weight(b) = \frac{t(b)}{n}$$

$$NValue = \sum_{b \in C_a} f_a(b) \times weight(b)$$

$$NCValue(a) = 0.8 \cdot CValue(a) + 0.2 \cdot NValue(a)$$

From the above formula, we find that $NCValue$ is mainly weighted by $CValue$. It treats the term candidate as a linguistic unit and evaluates its weight based on characteristics of the termhood, i.e. frequency and context word of the term candidate. The performance can be improved if feature measuring the adhesion of words within the term is incorporated.

3 Enhancement on Global TE: the *NTCValue*

Theoretically, the C/NC method can be improved by adding unithood feature to the term weighting formula. Based on the comparison of (Evert, S and B. Krenn, 2001), we explore T-Score, a competitive metric to evaluate the association between two words, as a unithood feature.

[2] Unithood refers to a degree of strength or stability of syntagmatic combinations or collocations.

[3] All experiments in this paper use the length of context is 3.

3.1 T-Score

The T-Score is used to measure the adhesion between two words in a corpus. It is defined by the following formula (Manning and Schuetze, 1999):

$$TS(w_i, w_j) = \frac{P(w_i, w_j) - P(w_i).P(w_j)}{\sqrt{\frac{P(w_i, w_j)}{N}}}$$

Where, $P(w_i, w_j)$ is the probability of bi-gram $w_i w_j$ in the corpus, $P(w)$ is the probability of word w in the corpus, and N is the total number of words in the corpus. The adhesion is a type of unithood feature since it is used to evaluate the intrinsic strength between two words of a term.

3.2 Incorporate T-Score within C/NC value

As discussed in 2.2, the most influential feature in the C/NC method is the term frequency. Our idea here is to combine the frequency with T-Score, a unithood feature. Taking the example in Table 1, the candidates have similar rank in the output using C/NC termhood approach.

massive tidal waves
gigantic tidal waves
killer tsunami tidal waves
deadly tidal waves
huge tidal waves
giant tidal waves
tsunamis tidal waves

Table 1. Example of similar terms [4]

To give better ranking and differentiation, we introduce T-Score to measure the adhesion between the words within the term. We use the minimum T-Score of all bi-grams in term a, $\min TS(a)$, as a weighted parameter for the term besides the term frequency. For a term $a = w_1.w_2...w_n$, the $\min TS(a)$ is defined as:

$$\min TS(a) = \min\{TS(w_i, w_{i+1})\}, i = 1...(n-1)$$

Term	$\min TS(\cdot)$
massive tidal waves	4.56
gigantic tidal waves	2.44
killer tsunami tidal waves	3.99
deadly tidal waves	3.15
huge tidal waves	2.20
giant tidal waves	1.35
***tsunamis tidal* waves**	**5.06**

Table 2. Term with Minimum T-Score value

Table 2 shows the $\min TS(a)$ of the different terms in table 1. Since $\min TS(a)$ can have a negative value, we only considered those terms with $\min TS(a) > 0$ and combined it with the term frequency. We redefine $CValue$ to $TCValue$ by replacing $f(a)$ using $F(a)$, as follows:

$$F(a) = \begin{cases} f(a) & \text{if } \min TS(a) \le 0 \\ f(a) \times \ln(2 + \min TS(a)) & \text{if } \min TS(a) > 0 \end{cases}$$

$$TCValue(a) = \log_2 |a| \cdot \left(F(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} F(b) \right)$$

The final weight, defined as *NTCValue*, is computed using the same parameter as $NCValue$.

$$NTCValue(a) = 0.8 \cdot TCValue(a) + 0.2 \cdot NValue(a)$$

4 Enhancement on Local Terms: Term Re-Extraction Method (TREM)

The extracted term candidates are ranked globally with best global terms promoted due to their distinguishing power. However, preliminary investigation on using linguistic patterns for extracting global term candidates for identifying term candidates of each document does not perform satisfactory, as high rank global terms do not reconcile well with the local term candidates identified using the linguistic patterns. A re-extraction process is thus evolved to derive local terms of a document from global terms using the $NTCValue$ of the global terms.

4.1 Local Term Candidate Extraction

A string (or term candidate) extracted based on linguistic pattern follows the maximum matching algorithm. As long as the longest string whose part-of-speech tag satisfies the linguistic pattern, it will be extracted. For this reason, some noises are extracted together with these candidates. Table 3 shows some examples of noisy term candidates.

Strait Times yesterday
THE World Cup
gross domestic product growth forecast
senior vice-president of DBS Vickers security on-line

Table 3. Examples of noisy candidates.

[4] The *italic* means a week adhesion.

Our intention here is to reduce the noise and also mine more good terms embedded within the noise by using the global terms. We favor recall over precision to get as many local terms as possible.

The examples in table 3 show the problem in detecting term candidate's boundary using linguistic patterns. The "Strait Times yesterday" is a bad term identified by linguistic patterns because all three words are tagged as "noun". The second one is caused by an error of the POS tagger. Because of capitalization, the word "THE" is being tagged wrongly as a "proper-noun" (NNP/NNPS), and not determiner (DT). Similarly, "gross domestic product growth forecast" and "senior vice-president of DBS Vickers security on-line" are complex noun-phrases that are not symbolized good terms in the document. The more expressive terms would be "gross domestic product", "DBS Vickers security", etc.

Our proposed algorithm utilizes the term weight from section 3.2 to do term re-extraction for each document through dynamic programming theory (Viterbi algorithm) to resolve the above problem.

4.2 Proposed algorithm

The algorithm for term re-extraction is outlined in **Figure 1**.

Algorithm: Term re-extraction for a document

Input: **L** ← global term list with $NTCValue$
T ← input for TREM $\mathrm{T} = w_1 w_2 ... w_n$

For $i = 2 \rightarrow n$
  If $\left(\mathrm{T}_{1,i} = w_1 ... w_i\right) \in \mathrm{L}$
    $MaxNTC(1, i) = NTC\left(\mathrm{T}_{1,i}\right)$
  Else $MaxNTC(1, i) = 0$
  End If
  For $j = 1 \rightarrow i - 1$
    If $\left(\mathbf{T}_{j+1,i} = w_{j+1} ... w_i\right) \in \mathbf{L}$
      $MaxNTC(1, i) = \max \left\{MaxNTC(1, j) + NTC\left(\mathbf{T}_{j+1,i}\right); MaxNTC(1, i)\right\}$
    End If
  End For
End For

Output: Updated term list for a document

Figure 1. Term Re-Extraction Algorithm

Where, $\mathrm{T}_{\mathrm{i,j}}$ is the word chain formed by the words from i to j of the term $\mathrm{T} = w_1 w_2 ... w_n$; $MaxNTC(1, i)$ is the maximum $NTCValue$ value from 1 to i of the term $\mathrm{T} = w_1 w_2 ... w_n$; and $NTC\left(\mathrm{T}_{1,i}\right)$ is the $NTCValue$ of $\mathrm{T}_{\mathrm{i,j}}$.

5 Experiments and Evaluations

5.1 Term Bank Collection

Term boundary is one of the main issues in terminology research. In our experiments, we consider a term based on the resources from Wikipedia. In each Wikipedia article, the editor annotated the key terminologies through the use of hyperlinks. We extracted the key terms for each article based on this markup. The entire Wikipedia contains about 1,910,974 English articles and 8,964,590 key terms. These terms are considered as Wikipedia term-bank and we use it to evaluate our performance. An extracted term is considered correct if and only if it is in the term-bank.

5.2 Corpus Collection

To evaluate the model, we use the corpus collected from Straits Times in year 2006. We separate the data into 12 months as showed in Table 4.

Month	Total articles	Total words
1	3,134	1,844,419
2	3,151	1,824,970
3	3,622	2,098,459
4	3,369	1,969,684
5	3,395	1,957,962
6	3,187	1,781,664
7	3,253	1,818,606
8	3,497	1,927,180
9	3,463	1,853,902
10	3,499	1,870,417
11	3,493	1,845,254
12	3,175	1,711,168

Table 4. Evaluation data from Straits Times.

5.3 NTCValue Evaluation

We evaluate the performance of global ranked terms using *average-precision*. A higher *average-precision* would mean that the list contains more good terms in higher rank. The average precision $\mathbf{Ave}P(.)$ of a term-list $L = \left\{t_1, t_2, ..., t_{|L|}\right\}$ with

L_c as the list of all correct terms in L $(L_c \subset L)$, is calculated by the following formula:

$$\mathbf{Ave}P(L) = \frac{1}{|L_c|} \sum_{1 \le k \le |L|} \left[r_k \times \left(\frac{1}{k} \sum_{1 \le i \le k} r_i \right) \right]$$

Where:

$$r_i = \begin{cases} 1 & t_i \in L_c \\ 0 & t_i \notin L_c \end{cases}$$

Table 5 shows the comparison result of the origin $NCValue$ and our $NTCValue$ on the ranking of global terms. The experiment is conducted on the data described in section 5.2. We evaluate the performance based on 8 different levels of top ranking terms.

Each cell in Table 5 contains a couple of $\mathbf{Ave}P(.)$ for $NCValue$ and $NTCValue$ $(NCValue / NTCValue)$ respectively. The $\mathbf{Ave}P(.)$ decreases gradually when we relax the threshold for the evaluation . The result shows that the term ranking using $NTCValue$ improves the performance significantly.

Number of top high term	01	02	03	04	05	06
50	0.70/0.77	0.57/0.81	0.52/0.80	0.51/0.78	0.55/0.80	0.67/0.69
100	0.60/0.73	0.59/0.77	0.51/0.79	0.50/0.74	0.57/0.78	0.64/0.70
200	0.55/0.70	0.56/0.75	0.53/0.78	0.49/0.72	0.55/0.77	0.62/0.69
500	0.53/0.67	0.54/0.70	0.54/0.71	0.48/0.68	0.53/0.71	0.57/0.65
1000	0.51/0.62	0.52/0.66	0.52/0.66	0.47/0.64	0.51/0.65	0.53/0.60
5000	0.48/0.58	0.49/0.61	0.49/0.62	0.45/0.60	0.49/0.61	0.49/0.56
10000	0.43/0.52	0.44/0.55	0.44/0.56	0.42/0.54	0.44/0.56	0.44/0.50
All_terms	0.38/0.47	0.39/0.49	0.40/0.50	0.37/0.48	0.39/0.49	0.38/0.45
Number of top high term	07	08	09	10	11	12
50	0.67/0.67	0.65/0.70	0.49/0.65	0.62/0.71	0.65/0.76	0.63/0.86
100	0.64/0.71	0.62/0.74	0.47/0.66	0.59/0.74	0.59/0.76	0.61/0.82
200	0.65/0.72	0.59/0.75	0.48/0.68	0.55/0.72	0.56/0.73	0.58/0.77
500	0.62/0.71	0.56/0.70	0.50/0.66	0.52/0.66	0.54/0.67	0.55/0.69
1000	0.59/0.66	0.54/0.66	0.50/0.64	0.49/0.64	0.51/0.64	0.54/0.65
5000	0.54/0.60	0.51/0.62	0.49/0.60	0.46/0.61	0.48/0.60	0.51/0.61
10000	0.46/0.53	0.46/0.55	0.45/0.55	0.43/0.56	0.44/0.55	0.46/0.55
All_terms	0.40/0.47	0.40/0.50	0.40/0.50	0.38/0.49	0.38/0.48	0.39/0.48

Table 5. Performance of NTCValue with C/NC value.

Method	Without TREM		TREM+NC		TREM+NTC	
Month	Precision	No. terms	Precision	No. terms	Precision	No. terms
1	44.98	23915	50.81	34910	50.85	34998
2	44.74	23772	50.22	34527	50.33	34657
3	44.39	28772	49.58	41691	49.59	41778
4	42.89	25857	48.78	38564	48.91	38589
5	44.67	25787	50.44	38252	50.38	38347
6	46.58	23293	51.80	33574	51.91	33651
7	46.35	23638	51.31	33990	51.35	34041
8	46.50	25869	51.91	37896	51.96	37973
9	46.16	25276	51.34	36632	51.39	36731
10	45.79	24987	50.99	36082	51.05	36179
11	45.28	24661	50.43	35894	50.54	35906
12	45.67	22745	50.73	32594	50.73	32673

Table 6. Term Re-Extraction evaluation result.

5.4 TREM Evaluation

We evaluate TREM based on the term bank described in section 5.1. Let M_i be the number of extracted terms for article i, N_i be the number of extracted terms in the term bank for article i, and n is the total articles in the test corpus. The accuracy is evaluated by the following formula:

$$P = \sum_{i=1}^{n} \frac{N_i}{M_i}$$

Table 6 shows the result of TREM. From the results, we can find that the accuracy has improved significantly after the re-extraction process. On top of that, the results of TREM based on $NTCValue$ is also slightly better than using $NCValue$. Moreover, the number of correct terms extracted by TREM using $NTCValue$ is higher than using $NCValue$.

6 Conclusions and Future Works

We introduce a term re-extraction process (TREM) using Viterbi algorithm to augment the local TE for each document in a corpus. The results in Table 6 show that TREM improves the precision of terms in local documents and also increases the number of correct terms extracted. We also propose a method to combine the C/NC value with T-Score. The results of our method, $NTCValue$, show that the motivation to combine the termhood features used in C/NC method, with T-Score, a unithood feature, improves the term ranking result. Results on Table 6 also show that $NTCValue$ gives a better result than the origin $NCValue$ for TREM.

In Table 5, the average scores for "All Term" are 38.8% and 48.3% for $NCValue$ and $NTCValue$ respectively. Therefore, NTCValue method improves global TE by 24.4% when compared to the origin NCValue method. With the same calculation, we also conclude that TREM outperforms the linguistic pattern method by 12% (*average scores are 50.7% and 45.3% for TREM and TREM-NTC respectively*).

In the future, we will focus on improving the performance of TREM by using more features, besides the weighting score.

References

C. Manning and H. Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press Cambridge, Massachusetts.

E. Milios, Y. Zhang, B. He, L. Dong. 2003. *Automatic Term Extraction and Document Similarity in Special Text Corpora*. Proceedings of the 6th Conference of the Pacific Association for Computational Linguistics (PACLing'03), Halifax, Nova Scotia, Canada, pp. 275-284.

Evert, S. and B. Krenn. 2001. *Methods for Qualitative Evaluation of Lexical Association Measures*. Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, pages 369 – 381.

Hideki Mima, Sophia Ananiadou. 2001. *An Application and Evaluation of the C/NC-Value Approach for the Automatic Term Recognition of Multi-Word Units in Japanese*. International Journal on Terminology.

Hiroshi Nakagawa, Tatsunori Mori. 2000. *Automatic Term Recognition based on Statistics of Compound Nouns*. Terminology, Vol.6, No.2, pp.195 – 210.

Hiroshi Nakagawa, Tatsunori Mori. 2002. *A Simple but Powerful Automatic Term Extraction Method*. 2nd International Workshop on Computational Terminology, ACL.

Katerine T. Frantzi, Sophia Ananiadou, and Junichi Tsujii. 1998. *The C-Value/NC-Value Method of Automatic Recognition for Multi-word terms*. Journal on Research and Advanced Technology for Digital Libraries.

Kyo Kageura. 1996. *Methods of Automatic Term Recognition - A Review*. Terminology, 3(2): 259 – 289, 1996.

Špela Vintar. 2004. *Comparative Evaluation of C-value in the Treatment of Nested Terms*. Memura 2004 – Methodologies and Evaluation of Multiword Units in Real-World Applications. Proceedings of the International Conference on Language Resources and Evaluation 2004, pp. 54-57.

Search Result Clustering Using Label Language Model

Yeha Lee Seung-Hoon Na Jong-Hyeok Lee
Div. of Electrical and Computer Engineering
Pohang University of Science and Technology (POSTECH)
Advanced Information Technology Research Center (AITrc)
San 31, Hyoja-Dong, Pohang, Republic of Korea, 790-784
{sion,nsh1979,jhlee}@postech.ac.kr

Abstract

Search results clustering helps users to browse the search results and locate what they are looking for. In the search result clustering, the label selection which annotates a meaningful phrase for each cluster becomes the most fundamental issue. In this paper, we present a new method of using the language modeling approach over Dmoz for label selection, namely label language model. Experimental results show that our method is helpful to obtain meaningful clustering labels of search results.

1 Introduction

Most contemporary search engines generate a long flat list in response to a user query. This result can be ranked by using criteria such as PageRank (Brin and Page, 1998) or relevancy to the query. However, this long flat list is uncomfortable to users, since it forces users to examine each page one by one, and to spend significant time and effort for finding the really relevant information. Most users only look into top 10 web pages in the list (Kummamuru et al., 2004). Thus many other relevant information can be missed out as a result. Clustering method is proposed in order to remedy the problem. Instead of the flat list, it groups the search results to clusters, and annotates a label with a representative words or phrases to each cluster. Then, these labeled clusters of search results are presented to users. Users can benefit from labeled clusters because the size of information presented is significantly reduced.

Search result clustering has several specific requirements that may not be required by other cluster algorithms. First, search result clustering should allow fast clustering and fast generation of a label on the fly, since it is an online process. This requirement can be met by adopting "snippets" [1] rather than entire documents of a search result set. Second, labels annotated for clusters should be meaningful to users because they are presented to users as a general view of results. For this reason, recent search result clustering researches focus on selecting meaningful labels. This differs from general clustering which focuses on the similarity of documents. In Zamir and Etzioni (Zamir and Etzioni, 1998), a few other key requirements of search result clustering are presented.

In this paper, we present a language modeling approach with Dmoz for search result clustering. Dmoz [2] is an Open Directory Project, and contains manually tagged categories for web-sites. Since these categories are built by human, they provide a good basis to build labels for clusters. We can view the problem of label selection for clusters as a problem of label generation by Dmoz.

We define a language model for each Dmoz-category and select labels for clusters according to the probability that this language model would generate candidate labels.

Thus, our method can select more meaningful labels for clusters because we use labels generated by human-tagged categories of Dmoz. The selected la-

[1] The term "snippet" is used here to denote fragment of a Web page returned by certain search engines

[2] Open Directory Project, http://www.dmoz.com/

bels enable users to quickly identify the desired information.

The paper is organized as follows. The next section introduces related works. In Section 3, we formulate the problem and show the detail of our approach. The experiment results and evaluations are presented in Section 4. Finally, we conclude the paper and discuss future works in Section 5.

2 RELATED WORKS

Many approaches have been suggested for organizing search results to improve browsing effectiveness. Previous researches such as scatter/Gather (Hearst and Pedersen, 1996) and Leuski (Leuski and Allan, 2000), Leouski (Leouski and Croft, 1996), cluster documents using document-similarity, and generate representative terms or phrases as labels. However, these labels are often not meaningful, which complicates user relevance judgment. They are also slow in generating clusters and labels because they use entire document contents in the process. Thus it is difficult to apply these approaches to search engine applications.

Due to the problems mentioned above, research in search result clustering has focused on choosing meaningful labels which is not usually addressed in general document clustering. Zeng et al. presented salient phrase ranking problem for label selection, which ranks labels scored by a combination of some properties of labels and documents (Zeng et al., 2004). Kummamuru regarded label selection as a problem making a taxonomy of the search result, and proposed a label selecting criterion based on taxonomy likelihood (Kummamuru et al., 2004). Zamir presented a Suffix Tree Clustering (STC) which identifies sets of documents that share common phrases, and clusters according to these phrases (Zamir and Etzioni, 1998). Maarek et al. and Osinski presented a singular value decomposition of the term-document matrix for search result clustering (Maarek et al., 2000), (Osinski and Weiss, 2004). The problem of these methods is that SVD is extremely time-consuming when applied to a large number of snippets. Ferragina proposed a method for generating hierarchical labels by which entire search results are hierarchically clustered (Ferragina and Gulli, 2005). This method produces a hierarchy of labeled clusters by constructing a sequence of labeled and weighted bipartite graphs representing the individual snippets on one side and a set of labeled clusters on the other side.

3 LABEL LANGUAGE MODEL

The main purpose of Label Language Model(LLM) is to generate meaningful labels on-the-fly from search results, specifically snippets, for web-users. The generated labels provide a view of the search result to users, and allow the users to navigate through them for their search needs.

Our algorithm is composed of the four phases:

1. Search result fetching
2. Candidate Labels Generation
3. Label Score Calculation
4. Post-processing

Search result fetching. LLM operates as a meta-search engine on top of established search engines. Our engine first retrieves results from dedicated search engines in response to user queries. The search results are parsed through HTML parser, and snippets are obtained as a result. We assume that these snippets contain enough information to provide user-relevance judgment. Hence, we can generate meaningful labels using only those snippets rather than the entire document contents of the search result set.

Candidate Labels Generation. Candidate labels are generated using the snippets obtained by search result fetching. Snippets are processed by Porter's algorithm for stemming and stopword removing, then every n-grams becomes a candidate label. Each candidate label is tagged with a score calculated by the Label Language Model. Finally, top N candidate labels with highest scores are displayed to users as labels for clusters of search result.

Label Score Calculation. Our model utilizes Dmoz to select meaningful labels. Dmoz is the largest, most comprehensive human-edited directory of the Web and classifies more than 3,500,000 sites in more than 460,000 categories. It is used for ranking and retrieval by many search engines, such as

Google (Rerragina and Gulli, 2005).

Language model ranks documents according to the probability that the language model of each document would generate the user query.

Dmoz is a human-edited directory, which contains meaningful categories. We can use the probability that categories of Dmoz would generate candidate labels as criteria to rank labels.

In our approach, the user query and the document correspond to the candidate label and the Dmoz's category, respectively. We can obtain the probability that LLM of each category would generate a label by language model. We assume that the probability of certain candidate label being generated can be estimated by the maximum value of the probability that LLM of each category would generate the candidate label.

Let $label_i$ be i^{th} label, w_{ij} be j^{th} word of $label_i$, and C_k be k^{th} category of Dmoz, respectively. If we assume that the labels are drawn independently from the distribution, then we can express the probability that Dmoz generates labels as follows:

$$p(label_i|Dmoz) = \max_k p(label_i|C_k) \quad (1)$$

$$p(label_i|C_k) = \prod p(w_{ij}|C_k) \quad (2)$$

We use two smoothing methods, Jelinek-Mercer smoothing and Dirichlet Priors smoothing (Zhai and Lafferty, 2001), in order to handle unseen words. The score of $label_i$ is calculated as follows:

$$S_i = \max_k \sum_j \log\left(1 + \frac{\lambda p(w_{ij}|C_k)}{(1-\lambda)p(w_{ij}|C_{all})}\right) \quad (3)$$

$$S_i = \max_k \sum_j \log \frac{\#(w_{ij}^k) + \mu p(w_{ij}|C_{all})}{\#(C_k) + \mu} \quad (4)$$

To solve the equation, $p(w_{ij}|C_k)$ and $p(w_{ij}|C_{all})$ should be estimated. Let $\#(C_k)$ and $\#(C_{all})$[3] be the number of words in k^{th} category and the number of words in Dmoz. Further, let $\#(w_{ij}^k)$ and $\#(w_{ij}^{all})$ be the number of word, w_{ij}, in k^{th} category and the number of word, w_{ij}, in Dmoz. Then $p(w_{ij}|C_k)$ is estimated as $\frac{\#(w_{ij}^k)}{\#(C_k)}$, and $p(w_{ij}|C_{all})$ as $\frac{\#(w_{ij}^{all})}{\#(C_{all})}$.

[3] C_{all} denotes all categories of Dmoz

In Candidate Labels Generation phase, all candidate labels are scored. After post-processing, candidate labels are shown in a descending order.

Post-processing. In post processing phase, labels are refined through several rules. First, labels composed of only query words are removed because they do not provide better clues for users. Second, labels that are contained in another label are removed. Since every possible n gram is eligible for candidate labels, multiple labels that differ only at the either ends, i.e., one label contained in another, can be assigned a high score. In such cases, longer labels are more specific and meaningful than shorter ones, therefore shorter ones are removed. Users can benefit from a more specific and meaningful label that clarifies what a cluster contains. Finally, Top N Labels with highest scores produced by post processing are presented to users.

4 EXPERIMENTS

We conducted several experiments with varying smoothing parameter values, λ, μ. We investigated the influence of the smoothing parameter on the label selection procedure.

4.1 Experiment Setup

Despite heavy researches on search result clustering, a standard test-set or evaluation measurement does not exist. This paper adopts the methodology of (Zeng et al., 2004) in order to evaluate the expressiveness of selected label and LLM

4.1.1 Test Data Set

We obtained Google's search results that correspond to fifty queries. The fifty queries are comprised of top 25 queries to Google and 25 from (Zeng et al., 2004). For each query of the fifty, 200 snippets from Google are obtained. Table 1 summarizes the query used in our experiment.

Search results obtained from Google are parsed to remove html-tag and stopword, and stemming is applied to obtain the snippets. Every n-gram of the snippets, where $n \leq 3$, becomes candidate labels. Labels that do not occur more than 3 times are removed from candidate set in order to reduce noise.

Type	Queries
2005 Google Top query	Myspace, Ares, Baidu, orkut, iTumes, Sky News, World of Warcraft, Green Day, Leonardo da Vinci, Janet Jackson, Hurricane Katrina, tsunami, xbox 360, Brad Pitt, Michael Jackson, American Idol, Britney Spears, Angelina Jolie, Harry Potter, ipod, digital camera, psp, laptop, computer desk
(Zeng et al., 2004) query	jaguar, apple, saturn, jobs, jordan, tiger, trec, ups, quotes, matrix, susan dumais, clinton, iraq, dell, disney, world war 2, ford, health, yellow pages, maps, flower, music, chat, games, radio, jokes, graphic design, resume, time zones, travel

Table 1: Queries used in experiment

4.1.2 Answer Label Set for Evaluation

In order to evaluate LLM, we manually created labels for each query which are desired as outputs of our test, and we refer them as answer labels. There might be a case where an answer label and label selected by our model are semantically equivalent but lexically different; for example, car and automobile. To mitigate the problem, we used Wordnet to handle two different words with the same semantic. We explain the use of Wordnet further in section 4.1.3.

4.1.3 Evaluation Measure & Method

We used precision at top N labels to evaluate the model. Precision at top N is defined as $P@N = \frac{M@N}{N}$, where is $M@N$ is the number of relevant labels among the top N generated labels to the answer set. As explained in section 4.1.2, the labels generated by our model might not be equal to answer labels even when they have the same semantic meaning. It might be very time consuming for a human to manually compare the two label set where one set can vary due to the varying smoothing parameter if semantic meaning also has to be considered.

We used WordNet's synonyms and hypernyms relationships in order to mitigate the problem addressed above. We regard a test label to be equal to an answer label when WordNet's synonyms or hypernyms relationship allows them. Only the first listed sense in Wordnet is used to prevent over-generation.

We evaluated the overall effectiveness of LLM with $P@N$ and the effect of smoothing parameter on $P@N$.

4.2 Experimental Result

We used $P@5$, $P@10$ and $P@20$ to evaluate the effectiveness of our model because most users disregard snippets beyond 20.

First, for each query, we obtained each label's MAP[4] for two smoothing methods. Figures 1 and 2 depicts MAP of Jelinke-Mercer smoothing and Dirichlet Priors smoothing.

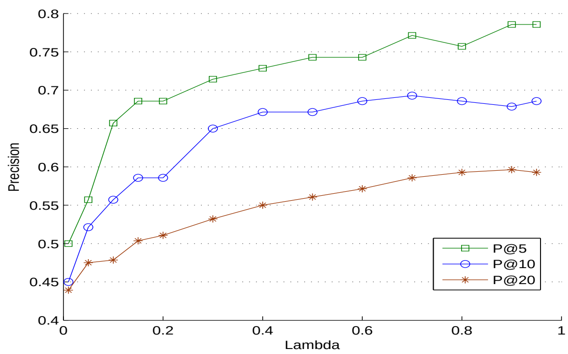


Figure 1: Jelinek-Mercer Smoothing

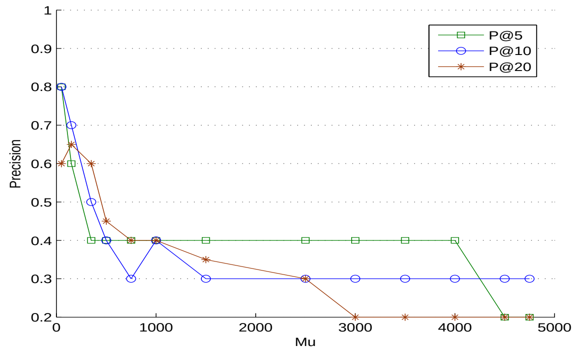


Figure 2: Dirichlet priors Smoothing

In figures 1 and 2, X-axis denotes smoothing parameter, and Y-axis denotes MAP. The figures show that the smaller the value of the smoothing is , the

[4] Mean Average Precision

higher the precision is. This indicates that a better label is selected when the probability that a specific category would generate the label is high. In our test result, when using Dirichlet smoothing, the precision of top 5 and 10 labels are 82% and 80%, thus users can benefit in browsing from our model using 5 or 10 labels. However, the precision rapidly drops to 60% at $P@20$. The low precision at $P@20$ shows the vulnerability of our model, indicating that our model needs a refinement.

Figure 3 shows individual precisions of labels for randomly selected five queries. The labels were generated by using Dirichlet priors smoothing.

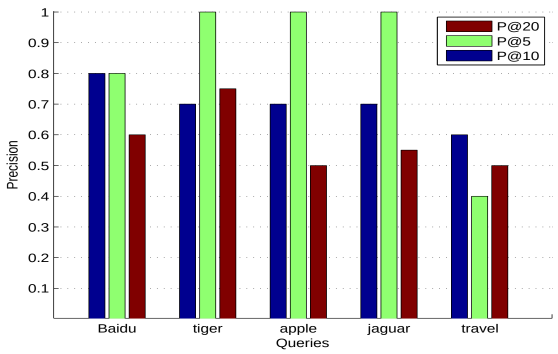


Figure 3: Using Dirichlet priors Smoothing

As shown in figure 1 and 2, the general order of result precisions is as follows: $P@20 \leq P@10 \leq P@5$. However, figure 3 shows that the precision for query "travel" is the lowest at $P@5$. This result indicates that words that appear many times in a specific category of Dmoz might have higher probability regardless of snippet's contents.

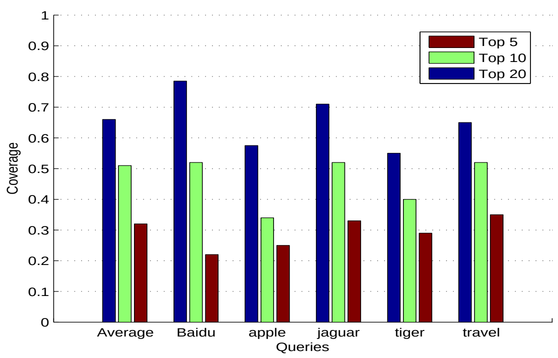


Figure 4: Coverage

Figure 4 shows the average coverage of labels generated by our model. The coverage of the labels is about 0.32%, 0.51% and 0.66% at top 5, 10 and 20 labels respectively. This means that the labels allow browsing over only 60% of the entire search results. The lack of coverage is another pitfall of our model, and further refining is needed.

Finally, in Table 2, we list top 10 labels for five queries.

5 CONCLUSION & FUTURE WORKS

We proposed a LLM for label selection of search results, and analyzed the smoothing parameter's effect on the label selection. Experimental results showed that LLM can pick up meaningful labels, and aid users in browsing web search results. Experimental results also validated our assumption that the high probability that Dmoz categories generate a label indicates meaningful labels. Further research directions remain as future works.

Our model is sensitive to Dmoz because we use the language model based on Dmoz. Our model may result in poor performance for labels that are not represented or over-represented in Dmoz. Therefore, it is meaningful to study how sensitive to Dmoz the performance of the LLM is, and how to mitigate sensitivity. We used Google's search results as an input to our system. However, multiple engines offer a better coverage of the web because of the low overlap of current search engines (Bharat and Broder, 1998). Further work can utilize multiple engines to generate input to our system. In our test, snippet's title and content were assigned the same weight, and titles and descriptions of Dmoz's category were also assigned the same weight. Future work might benefit from varying the weights to them. We did not utilize the information buried in the documents, such as $tf \cdot idf$, but used only knowledge provided by the external system, Dmoz. We believe that this also affected LLM's poor performance on over-represented terms. Future work will benefit from incorporating the information derivable from the documents.

6 ACKNOWLEDGMENTS

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center

Queries	Labels
Baidu	language search set, Chinese search engine, search engine company, Baidu.com, MP3 Search, Baidu engine, Japanese Search Engine, IPO, search market, Mobile
apple	Mac OS X, iPod, Apple Macintosh, Apple products, language character set, Music Store, Apple develops, Apple Support, information, San Francisco
jaguar	Mac OS X, Jaguar Cars, Land Rover, Jaguar XJ, Jaguar XK, largest cat, Leopard, Photos tagged jaguar, Jaguar dealer, Jaguar Clubs
tiger	Mac OS X, Tiger Woods, Tiger Cats, Detroit Tigers, Security tool, Parts PC Components, Paper Tiger, Adventure Tour, National Zoo, Tiger Beat
travel	Car Rental, airline tickets, discount hotels, Plan trip, Airfares, package holidays, Visa, Travel Cheap, Destination guides, Travel news

Table 2: Queries used in experiment

(AITrc), also in part by the BK 21 Project and MIC & IITA through IT Leading R&D Support Project in 2007.

References

P. Rerragina and A. Gulli. 2005. A personalized search engine based on web-snippet hierarchical clustering. In *Special Interest Tracks and Poster Proceedings of WWW-05, International Conference on the World Wide Web,* 801-810

H. Zeng, Q. He, Z. Chen, W. Ma and J. Ma 2004. Learning to cluster web search results. In *Proceedings of the 27th ACM SIGIR Conference on Research and Development of Information Retrieval*

M. A. Hearst and J. O. Pedersen. 1996. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of 19th ACM SIGIR Conference on Research and Development in Information Retrieval*, 76-84

K. Kummamuru, R. Lotlikar, S. Roy, K. Signal and R. Krishnapuram 2004. A hierarchical monothetic document clustering algorithm for summarization browsing search results. In *Proceedings of 13th International Conference on World Wide Web,* 658-665

A. Leuski and J. Allan. 2000. Improving Interactive Retrieval by Combining Ranked List and Clustering. In *Proceedings of RIAOI, College de France*, 665-681

A. V. Leouski and W. B. Croft. 1996. An Evaluation of Techniques for Clustering Search Results. In *Technical Report IR-76,* Department of Computer Science, University of Massachusetts, Amherst

O. Zamir and O. Etzioni. 1998. Web Document Clustering: A Feasibility Demonstration. In *Proceedings of the 21th ACM SIGIR Conference on Research and Development of Information Retrieval*, 46-54

Y. Maarek, R. Fagin, I. Ben-Shaul and D. Pelleg. 2000. Ephemeral document clustering for Web applications. *Technical Report RJ 10186, IBM, San Jose, US*

S. Osinski and D. Weiss. 2004. Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data In *Proceedings of IIPWM-04, 5th Conference on Intelligent Information Processing and Web Mining,* 369-377

P. Ferragina and A. Gulli. 2005. A personalized search engine based on Web-snippet hierarchical clustering. In *Special Interest Tracks and Poster Proceedings of WWW-05, International conference on the World Wide Web,* 801-810

S. Brin and L. Page 1998. The anatomy of a large-scale hypertextual(Web) Search Engine. In *Proceedings of the 7th International Conference on World Wide Web,* 107-117

C. Zhai and J. Lafferty 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th ACM SIGIR Conference on Research and Development of Information Retrieval*, 334-342

K. Bharat and A. Broder. 1998. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the 7th International Conference on World Wide Web*

Effects of Related Term Extraction in Transliteration into Chinese

HaiXiang Huang **Atsushi Fujii**
Graduate School of Library, Information and Media Studies
University of Tsukuba
1-2 Kasuga, Tsukuba, 305-8550, Japan
{lectas21,fujii}@slis.tsukuba.ac.jp

Abstract

To transliterate foreign words, in Japanese and Korean, phonograms, such as Katakana and Hangul, are used. In Chinese, the pronunciation of a source word is spelled out using Kanji characters. Because Kanji is ideogrammatic representation, different Kanji characters are associated with the same pronunciation, but can potentially convey different meanings and impressions. To select appropriate Kanji characters, an existing method requests the user to provide one or more related terms for a source word, which is time-consuming and expensive. In this paper, to reduce this human effort, we use the World Wide Web to extract related terms for source words. We show the effectiveness of our method experimentally.

1 Introduction

Reflecting the rapid growth of science, technology, and economies, new technical terms and product names have progressively been created. These new words have also been imported into different languages. There are two fundamental methods for importing foreign words into a language.

In the first method–*translation*–the meaning of the source word in question is represented by an existing or new word in the target language.

In the second method–*transliteration*–the pronunciation of the source word is represented by using the phonetic alphabet of the target language, such as Katakana in Japanese and Hangul in Korean. Technical terms and proper nouns are often transliterated.

In Chinese, Kanji is used to spell out both conventional Chinese words and foreign words. Because Kanji is ideogrammatic, an individual pronunciation can be represented by more than one character. If several Kanji strings are related to the same pronunciation of the source word, their meanings will be different and convey different impressions.

For example, “Coca-Cola” can be represented by different Kanji strings in Chinese with similar pronunciations, such as “可口可乐” and “口卡口拉”. The official transliteration is “可口可乐”, which comprises “可口 (tasty)” and “可乐 (pleasant)”, and is therefore associated with a positive connotation. However, “口卡口拉” is associated with a negative connotation because this word includes “口卡 ”, which is associated with “choking”.

For another example, the official transliteration of the musician Chopin’s name in Chinese is “肖邦”, where “肖” is commonly used for Chinese family names. Other Kanji characters with the same pronunciation as “肖” include “消”. However, “消”, which means “to disappear”, is not ideal for a person’s name.

Thus, Kanji characters must be selected carefully during transliteration into Chinese. This is especially important when foreign companies intend to introduce their names and products into China.

In a broad sense, the term “transliteration” has been used to refer to two tasks. The first task is transliteration in the strict sense, which creates new words in a target language (Haizhou et al., 2004; Wan and Verspoor, 1998; Xu et al., 2006). The second task is back-transliteration (Knight and Graehl, 1998), which identifies the source word correspond-

ing to an existing transliterated word. Both tasks require methods that model pronunciation in the source and target languages.

However, by definition, in back-transliteration, the word in question has already been transliterated and the meaning or impression of the source word does not have to be considered. Thus, back-transliteration is outside the scope of this paper. In the following, we use the term "transliteration" to refer to transliteration in the strict sense.

Existing transliteration methods for Chinese (Haizhou et al., 2004; Wan and Verspoor, 1998), which aim to spell out foreign names of people and places, do not model the impression the transliterated word might have on the reader.

Xu et al. (2006) proposed a method to model both the impression and the pronunciation for transliteration into Chinese. In this method, impression keywords that are related to the source word are used. However, a user must provide impression keywords, which is time-consuming and expensive.

In this paper, to reduce the amount of human effort, we propose a method that uses the World Wide Web to extract related terms for source words.

2 Overview

Figure 1 shows our transliteration method, which models pronunciation, impression, and target language when transliterating foreign words into Chinese. Figure 1 is an extension of the method proposed by Xu et al. (2006) and the part surrounded by a dotted line is the scheme we propose in this paper. We will explain the entire process using Figure 1.

There are two parts to the input for our method. First, a source word to be transliterated into Chinese is requested. Second, the category of the source word, such as "company" or "person", is requested. The output is one or more Kanji strings.

Using the pronunciation model, the source word is converted into a set of Kanji strings whose pronunciation is similar to that of the source word. Each of these Kanji strings is a transliteration candidate.

Currently, we use Japanese Katakana words as source words, because Katakana words can be easily converted into pronunciations using the Latin alphabet. In Figure 1, the Katakana word "epuson (EPSON)" is used as an example source word. However, in principle, any language that uses a phonetic script can be a source language for our method.

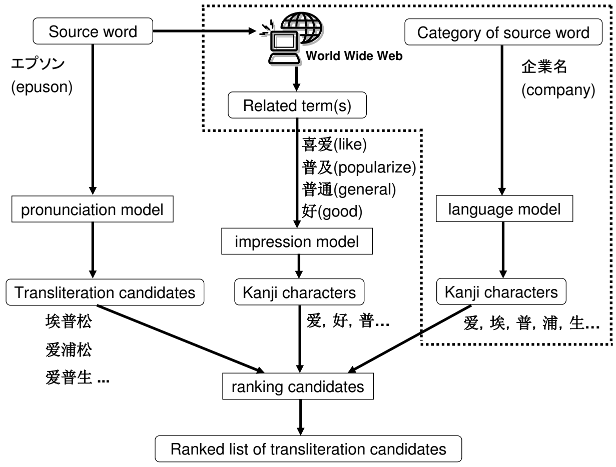


Figure 1: Overview of our transliteration method.

Using the impression model, one or more related terms are converted into a set of Kanji characters. In Xu et al. (2006), one or more words that describe the impression of the source word are used as related terms (i.e., impression keywords). Because impression keywords are given manually, users must have a good command of Chinese. In addition, the task of providing impression keywords is expensive. We solve these problems by automatically extracting terms related to the source word from the Web.

Unlike Xu at al. (2006), the language model for the category of the source word is used. For example, if the category is "person", Kanji characters that are often used for personal names in Chinese are preferably used for the transliteration.

Because of the potentially large number of selected candidates, we need to rank the candidates. We model pronunciation, impression, and target language in a probabilistic framework, so that candidates are sorted according to their probability score. In practice, the Kanji characters derived via the impression and language models are used to re-rank the candidates derived via the pronunciation model.

3 Probabilistic Transliteration Model

Given a romanized source word R, a set of related terms W, and the category of the source word C, our purpose is to select the Kanji string K that maximizes $P(K|R, W, C)$, which is evaluated as shown in Equation (1), using Bayes's theorem.

$$
\begin{aligned}
&P(K|R,W,C)\\
&= \frac{P(R,W,C|K)\times P(K)}{P(R,W,C)}\\
&\approx \frac{P(R|K)\times P(W|K)\times P(C|K)\times P(K)}{P(R,W,C)} \quad (1)\\
&\propto P(R|K)\times P(W|K)\times P(C|K)\times P(K)\\
&= P(R|K)\times P(W|K)\times P(C,K)
\end{aligned}
$$

Xu et al. (2006) did not consider the category of the source word and computed $P(K|R,W)$.

In the third line of Equation (1), we assume the conditional independence of R, W, and C given K. In the fourth line, we omit $P(R,W,C)$, which is independent of K. This does not affect the relative rank of Kanji strings, when ranked in terms of $P(K|R,W,C)$. If a user intends to select more than one Kanji string, those Ks associated with higher probabilities should be selected. In Figure 1, R, W, and C are "epuson", "喜爱 普及 普通 生动" and "企業名", respectively, and a K is "埃普松".

In Equation (1), $P(K|R,W,C)$ can be approximated by the product of $P(R|K)$, $P(W|K)$, and $P(C,K)$. We call these three factors the pronunciation, impression, and language models, respectively.

The implementation of $P(R|K)$ and $P(W|K)$ is the same as in Xu et al. (2006). While $P(R|K)$ has commonly been used in the literature, the basis of $P(W|K)$ should perhaps be explained. $P(W|K)$ is computed using co-occurrence frequencies of each word in W and each character in K, for which we extracted co-occurrences of a word and a Kanji character from a dictionary of Kanji in Chinese. Please see Xu et al. (2006) for details. However, unlike Xu et al. (2006), in which W was provided manually, we automatically extract W from the Web.

While Xu et al. (2006) did not use the language model, we compute $P(C,K)$ by Equation (2).

$$P(C,K) = P(C)\times P(K|C) \propto P(K|C) \quad (2)$$

We omit $P(C)$, which is independent of K. Thus, we compute $P(K|C)$, which is the probability that a Kanji string K is selected given category C.

To compute $P(K|C)$, we decompose K into single Kanji characters. We used a character unigram model and produced the following three language models.

- general model: one month of newspaper articles in the PFR corpus[1] were used. In this model, 4 540 character types (12 229 563 tokens) are modeled.
- company model: a list of 22 569 company names in CNLP (Chinese Natural Language Processing)[2] was used. In this model, 2 167 character types (78 432 tokens) are modeled.
- person model: a list of 38 406 personal names in CNLP was used. In this model, 2 318 character types (104 443 tokens) are modeled.

To extract Kanji characters from the above corpus and lists, we performed morphological analysis by SuperMorpho[3] and removed functional words and symbols. While the general model is not adapted to any specific category, the other models are adapted to the company and person categories, respectively. Although the effect of adapting language models has been explored in spoken language processing, no attempt has been made for transliteration.

4 Extracting Related Terms

To extract related terms for a source word, we used Wikipedia[4], which is a free encyclopedia on the Web and includes general words, persons, places, companies, and products, as headwords. We extracted related term candidates for a source word as follows.

1. We consulted the Japanese Wikipedia for the source word and obtained the result page.
2. We deleted HTML tags from the result page and performed morphological analysis by ChaSen[5].
3. We extracted nouns and adjectives as related term candidates.

We used mutual information (Turney, 2001) to measure the degree of relation between the source word and a related term candidate by Equation (3).

$$I(X,Y) = \log \frac{P(X,Y)}{P(X)\times P(Y)} \quad (3)$$

[1]http://icl.pky.edu.cn/
[2]http://www.nlp.org.cn/
[3]http://www.omronsoft.com/
[4]http://ja.wikipedia.org/wiki/
[5]http://chasen.naist.jp/hiki/ChaSen/

X and Y denote the source word and a related term candidate, respectively. $P(X)$ and $P(Y)$ denote probabilities of X and Y, respectively. $P(X, Y)$ denotes the joint probability of X and Y.

To estimate the above three probabilities, we followed the method proposed by Turney (2001). We used the Yahoo!JAPAN[6] search engine and replaced $P(A)$ in Equation (3) with the number of pages retrieved by the query A. Here, "A" can be "X", "Y", or "X and Y". Then, we selected up to 10 Ys with the greatest $I(X, Y)$ and translated them into Chinese using the Yahoo!JAPAN machine translation system.

Table 1 shows examples of related terms for the source word "ミサ (mass)", such as "典礼 (ceremony)" and " 奉献 (dedication)". Irrelevant candidates, such as " 会 (meeting)" and "こと (thing)", were discarded successfully.

Table 1: Example of related terms for "ミサ (mass)".

Extracted related terms		Discarded candidates	
Japanese	English	Japanese	English
典礼	ceremony	会	meeting
奉献	dedication	こと	thing
司教	bishop	会議	meeting
教会	church	参加	join

5 Experiments

5.1 Method

To evaluate the effectiveness of the related term extraction in the transliteration, we compared the accuracy of the following three methods.

- A combination of the pronunciation and language models that does not use the impression model, $P(W|K)$, in Equation (1),
- Our method, which uses Equation (1) and uses automatically extracted related terms as W,
- Equation (1), in which manually provided impression keywords are used as W.

To make the difference between the second and third methods clear, we use the terms "related term (RT)" and "impression keyword (IK)" to refer to words provided automatically and manually, respectively. Then, we call the above three methods "PL", "PL+RT", and "PL+IK", respectively. PL and PL+IK are the lower bound and the upper bound of the expected accuracy, respectively. PL+IK is the same as in Xu et al. (2006), but the language model is adapted to the category of source words.

To produce test words for the transliteration, we first collected 210 Katakana words from a Japanese–Chinese dictionary. These 210 words were also used by Xu et al. (2006) for experiments. We then consulted Wikipedia for each of the 210 words and selected 128 words that were headwords in Wikipedia, as test words. Details of the 128 test words are shown in Table 2.

Table 2: Categories of test words.

Category	#Words	Example word		
		Japanese	Chinese	English
General	24	エンジェル	安琪儿	angel
Company	35	インテル	英特尔	Intel
Product	27	アウディ	奥迪	Audi
Person	13	ショパン	肖邦	Chopin
Place	29	オハイオ	俄亥俄	Ohio

We selectively used the three language models explained in Section 3. We used the general model for general words. We used the company model for company and product names, and used the person model for person and place names. A preliminary study showed that the language model adaptation was generally effective for transliteration. However, because the focus of this paper is the related term extraction, we do not describe the evaluation of the language model adaptation.

Two Chinese graduate students who had a good command of Japanese served as assessors and produced reference data, which consisted of impression keywords used for PL+IK and correct answers for the transliteration. Neither of the assessors was an author of this paper. The assessors performed the same task for the 128 test words independently, to enhance the objectivity of the evaluation.

We produced the reference data via the following procedure that is the same as that of Xu et al. (2006).

First, for each test word, each assessor provided one or more impression keywords in Chinese. We did not restrict the number of impression key-

[6] http://www.yahoo.co.jp/

words per test word; the number was determined by each assessor. We provided the assessors with the descriptions for the test words from the source Japanese–Chinese dictionary, so that the assessors could understand the meaning of each test word.

Second, for each test word, we applied the three methods (PL, PL+RT, and PL+IK) independently, which produced three lists of ranked candidates.

Third, for each test word, each assessor identified one or more correct transliterations, according to their impression of the test word. It was important not to reveal to the assessors which method produced which candidates. By these means, we selected the top 100 transliteration candidates from the three ranked lists. We merged these candidates, removed duplications, and sorted the remaining candidates by character code. The assessors judged the correctness of up to 300 candidates for each test word. The average number of candidates was $36\,976$.

The resultant reference data were used to evaluate the accuracy of each method in ranking transliteration candidates. We used the average rank of correct answers in the list as the evaluation measure. If more than one correct answer was found for a single test word, we first averaged the ranks of these answers and then averaged the ranks over the test words.

For each test word, there was more than one type of "correct answer", as follows:

(a) transliteration candidates judged as correct by either of the assessors independently,

(b) transliteration candidates judged as correct by both assessors,

(c) transliteration defined in the source Japanese–Chinese dictionary.

In (a), the coverage of correct answers is the largest, whereas the objectivity of the judgment is the lowest. In (c), the objectivity of the judgment is the largest, whereas the coverage of correct answers is the lowest. In (b), where the assessors did not disagree about the correctness, the coverage of the correctness and the objectivity are in between.

The number of test words was 128 for both (a) and (c), but 76 for (b). The average numbers of correct answers were 1.65, 1.04, and 1 for (a), (b), and (c), respectively.

5.2 Results and Analyses

Table 3 shows the average rank of correct answers for different cases. Looking at Table 3, for certain categories, such as "Place", when the impression model was used, the average rank was low. However, on average, the average rank for PL+RT was lower than that for PL+IK, but was higher than that for PL, irrespective of the answer type.

Figures 2 and 3 show the distribution of correct answers for different ranges of ranks, using answer types (a) and (c) in Table 3, respectively. Because the results for types (a) and (b) were similar, we show only the results of type (a), for the sake of conciseness. In Figure 2, the number of correct answers in the top 10 for PL+RT was smaller than that for PL+IK, but was greater than that for PL.

In Figure 3, the number of correct answers in the top 10 for PL+RT was greater than those for PL and PL+IK. Because in Figure 3, the correct answers were defined in the dictionary and were independent of the assessor judgments, PL+IK was not as effective as in Figure 2.

In summary, the use of automatically extracted related terms was more effective than the method that does not use the impression model. We also reduced the manual cost of providing impression keywords, while maintaining the transliteration accuracy.

Table 4 shows examples of related terms or impression keywords for answer type (c). In Table 4, the column "Rank" denotes the average rank of correct answers for PL+RT and PL+IK, respectively. For "ミサ (mass)", the rank for PL+RT was higher than that for PL+IK. However, for "カタール (the State of Qatar)", the rank for PL+RT was lower than that for PL+IK. One reason for this is that most related terms for PL+RT were names of countries that border Qatar, which do not describe Qatar well, compared with impression keywords for PL+IK, such as "沙漠 (desert)" and " 石油 (oil)". This example indicates room for improvement in the related term extraction algorithm.

6 Conclusion

For transliterating foreign words into Chinese, the pronunciation of a source word is spelled out with Kanji characters. Because Kanji is an ideogrammatic script, different Kanji characters are associ-

Table 3: Average rank of correct answers for different methods in different cases.

Category	Answer type (a)			Answer type (b)			Answer type (c)		
	PL	PL+RT	PL+IK	PL	PL+RT	PL+IK	PL	PL+RT	PL+IK
General	189	165	167	44	49	52	84	61	65
Company	232	208	203	33	29	27	317	391	325
Product	197	175	166	34	27	21	313	198	198
Person	98	69	44	4	4	4	114	154	75
Place	85	133	95	13	14	16	76	98	89
Avg.	160	150	135	26	25	24	181	160	150

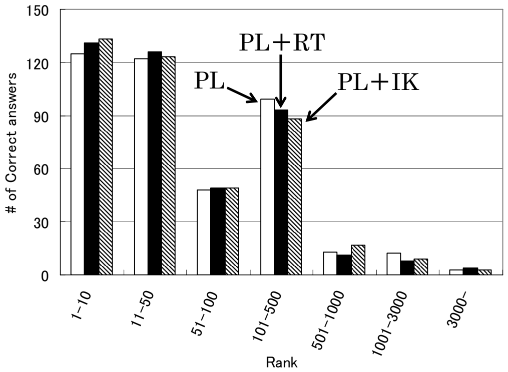


Figure 2: Rank for correct answer type (a).

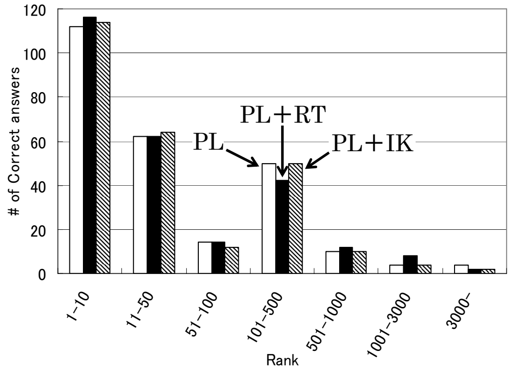


Figure 3: Rank for correct answer type (c).

Table 4: Examples of related terms and impression keywords used for experiments.

Source word	Answer	Method	Rank	Examples of related terms or impression keywords
ミサ (mass)	弥撒	PL+RT	8	典礼 (ceremony), 主教 (bishop), 奉献 (dedication), 教会 (church)
		PL+IK	10	典礼 (ceremony), 主教 (bishop), 信仰 (belief), 教会 (church)
カタール (State of Qatar)	卡塔尔	PL+RT	103	科威特 (State of Kuwait), 也门 (Republic of Yemen)
		PL+IK	61	阿拉伯 (Arab), 沙漠 (desert), 石油 (oil), 干燥 (dryness)

ated with the same pronunciation, but can potentially convey different meanings and impressions. In this paper, to select appropriate characters for transliterating into Chinese, we automatically extracted related terms for source words using the Web. We showed the effectiveness of our method experimentally.

References

Li Haizhou, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Peter D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning*, pages 419–502.

Stephen Wan and Cornelia Maria Verspoor. 1998. Automatic English-Chinese name transliteration for development of multilingual resources. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 1352–1356.

LiLi Xu, Atsushi Fujii, and Tetsuya Ishikawa. 2006. Modeling Impression in Probabilistic Transliteration into Chinese. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 242–249.

A Structured Prediction Approach for Statistical Machine Translation

Dakun Zhang* **Le Sun†** **Wenbo Li***

*Institute of Software, Graduate University
Chinese Academy of Sciences
Beijing, China, 100080
{dakun04,liwenbo02}@iscas.cn

†Institute of Software
Chinese Academy of Sciences
Beijing, China, 100080
sunle@iscas.cn

Abstract

We propose a new formally syntax-based method for statistical machine translation. Transductions between parsing trees are transformed into a problem of sequence tagging, which is then tackled by a search-based structured prediction method. This allows us to automatically acquire translation knowledge from a parallel corpus without the need of complex linguistic parsing. This method can achieve comparable results with phrase-based method (like Pharaoh), however, only about ten percent number of translation table is used. Experiments show that the structured prediction approach for SMT is promising for its strong ability at combining words.

1 Introduction

Statistical Machine Translation (SMT) is attracting more attentions than rule-based and example-based methods because of the availability of large training corpora and automatic techniques. However, rich language structure is difficult to be integrated in the current SMT framework. Most of the SMT approaches integrating syntactic structures are based on probabilistic tree transducers (tree-to-tree model). This leads to a large increase in the model complexity (Yamada and Knight 2001; Yamada and Knight 2002; Gildea 2003; Galley et al. 2004; Knight and Graehl 2005; Liu et al. 2006). However, formally syntax-based methods propose simple but efficient ways to parse and translate sentences (Wu 1997; Chiang 2005).

In this paper, we propose a new model of SMT by using structured prediction to perform tree-to-tree transductions. This model is inspired by Sagae and Lavie (2005), in which a stack-based representation of monolingual parsing trees is used. Our contributions lie in the extension of this representation to bilingual parsing trees based on ITGs and in the use of a structured prediction method, called SEARN (Daumé III et al. 2007), to predict parsing structures.

Furthermore, in order to facilitate the use of structured prediction method, we perform another transformation from ITG-like trees to label sequence with the grouping of stack operations. Then the structure preserving problem in translation is transferred to a structured prediction one tackled by sequence labeling method such as in Part-of-Speech (POS) tagging. This transformation can be performed automatically without complex linguistic information. At last, a modified search process integrating structure information is performed to produce sentence translation. Figure 1 illustrates the process flow of our model. Besides, the phrase extraction is constrained by ITGs. Therefore, in this model, most units are word based except that we regard those complex word alignments as a whole (i.e. phrase) for the simplicity of ITG-like tree representations.

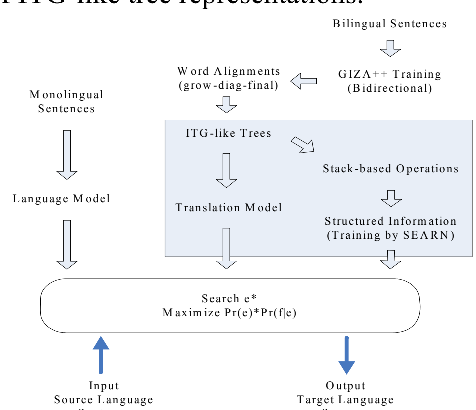


Figure 1: Chart of model framework

The paper is organized as follows: related work is show in section 2. The details of the transforma-

tion from word alignments to structured parsing trees and then to label sequence are given in section 3. The structured prediction method is described in section 4. In section 5, a beam search decoder with structured information is described. Experiments are given for three European language pairs in section 6 and we conclude our paper with some discussions.

2 Related Work

This method is similar to block-orientation modeling (Tillmann and Zhang 2005) and maximum entropy based phrase reordering model (Xiong et al. 2006), in which local orientations (left/right) of phrase pairs (blocks) are learned via MaxEnt classifiers. However, we assign shift/reduce labeling of ITGs taken from the shift-reduce parsing, and classifier is learned via SEARN. This paper is more elaborated by assigning detailed stack-operations.

The use of structured prediction to SMT is also investigated by (Liang et al. 2006; Tillmann and Zhang 2006; Watanabe et al. 2007). In contrast, we use SEARN to estimate one bilingual parsing tree for each sentence pair from its word correspondences. As a consequence, the generation of target language sentences is assisted by this structured information.

Turian et al. (2006) propose a purely discriminative learning method for parsing and translation with tree structured models. The word alignments and English parse tree were fed into the GenPar system (Burbank et al. 2005) to produce binarized tree alignments. In our method, we predict tree structures from word alignments through several transformations without involving parser and/or tree alignments.

3 Transformation

3.1 Word Alignments and ITG-like Tree

First, following Koehn et al. (2003), bilingual sentences are trained by GIZA++ (Och and Ney 2003) in two directions (from source to target and target to source). Then, two resulting alignments are recombined to form a whole according to heuristic rules, e.g. grow-diag-final. Second, based on the word alignment matrix, one unique parsing tree can be generated according to ITG constraints where the "left-first" constraint is posed. That is to say, we always make the leaf nodes as the right sons as possible as they can. Here we present two basic operations for mapping tree items, one is in order and the other is in reverse order (see Figure 2). Basic word alignments are in (a), while (b) is their corresponding alignment matrix. They can be described using ITG-like trees (c).

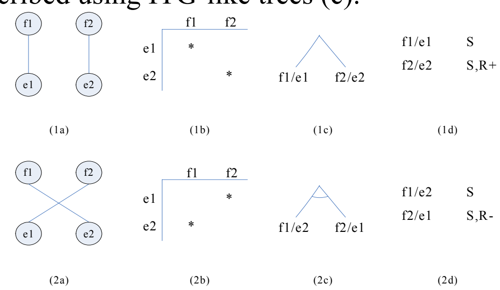


Figure 2: Two basic representations for tree items

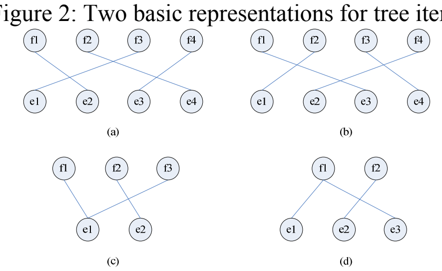


Figure 3: "inside-out" transpositions (a) and (b) with two typical complex sequences (c) and (d). In (c) and (d), word correspondence f2-e2 is also extracted as sub-alignments.

The two widely known situations that cannot be described by ITGs are called "inside-out" transpositions (Figure 3 a & b). Since they cannot be decomposed in ITGs, we consider them as basic units. In this case, phrase alignment is used. In our model, more complex situations exist for the word correspondences are generated automatically from GIZA++. At the same time, we also keep the sub-alignments in those complex situations in order to extend the coverage of translation options. The sub-alignments are restricted to those that can be described by the two basic operations. In other words, for our ITG-like tree, the nodes are mostly word pairs, except some indecomposable word sequences pairs. Figure 3 shows four typical complex sequences viewed as phrases.

Therefore, our ITG-like trees take some phrase alignments into consideration and we also keep the sub-alignments in these situations. Tree items in our model are restricted to minimum constituents for the simplicity of parsing tree generation. Then we extract those word pairs from tree items, instead of all the possible word sequences, as our translation table. In this way, we can greatly reduce the number of translation pairs to be consideration.

3.2 SHIFT and REDUCE Operations

Sagae and Lavie (2005) propose a constituency-based parsing method to determine sentence dependency structures. This method is simple and efficient, which makes use of SHIFT and REDUCE operations within a stack framework. This kind of representations can be easily learned by a classifier with linear time complexity.

In their method, they build a parse tree of a sentence one word at a time just as in a stack parser. At any time step, they either shift a new word on to the stack, or reduce the top two elements on the stack into a new non-terminal.

Sagae and Lavie's algorithms are designed for monolingual parsing problem. We extend it to represent our ITG-like tree. In our problem, each word pairs can be viewed as tree **items** (nodes). To handle our tree alignment problem, we need to define two REDUCE operations: REDUCE in order and REDUCE in reverse order. We define these three basic operations as follows:

- S: SHIFT - push the current item onto the stack.
- R+: REDUCE in order - pop the first two items from the stack, and combine them in the **original order** on the target side, then push back.
- R-: REDUCE in reverse order - pop the first two items from the stack, and combine them in the **reverse order** on the target side, then push back.

Using these operators, our ITG-like tree is transformed to serial stack operations. In Figure 2, (d) is such a representation for the two basic alignments. Therefore, the structure of word aligned sentences can be transformed to an operation sequence, which represents the bilingual parsing correspondences.

After that, we attach these operations to each corresponding tree item like a sequence labeling problem. We need to perform another "grouping" step to make sure only one operation is assigned to each item, such as "S,R+", "S,R-,R+", etc. Then, those grouped operations are regarded as a whole and performed as one label. The number of this kind of labels is decided by the training corpus[1]. Having defined such labels, the prediction of tree structures is transformed to a label prediction one. That is, giving word pairs as input, we transform them to their corresponding labels (stack operations) in the output. At the same time, tree transductions are encoded in those labels. Once all the "labels" are performed, there should be only one element in the stack, i.e. the generating sentence translation pairs. See Appendix A for a more complete example in Chinese-English with our defined operations.

Another constraint we impose is to keep the least number of elements in stack at any time. If two elements on the top of the stack can be combined, we combine them to form a single item. This constraint can avoid having too many possible operations for the last word pair, which may make future predictions difficult.

4 Structured Prediction

SEARN is a machine learning method proposed recently by Daumé III et al. (2007) to solve structured prediction problems. It can produce a high prediction performance without compromising speed, simplicity and generality. By incorporating the search and learning process, SEARN can solve the complex problems without having to perform explicit decoding any more.

In most cases, a prediction of input x in domain X into output y in domain Y, like SVM and decision trees, cannot keep the structure information during prediction. SEARN considers this problem as a cost sensitive classification one. By defining features and a loss function, it performs a cost sensitive learning algorithm to learn predictions. During each iteration, the optimal policy (decided by previous classifiers) generates new training examples through the search space. These data are used to adjust performance for next classifier. Then, iterations can keep this algorithm to perform better for prediction tasks. Structures are preserved for it integrates searching and learning at the same time.

4.1 Parsing Tree Prediction

For our problem, using SEARN to predict the stack-based ITG-like trees, given word alignments as input, can benefit from the advantages of this algorithm. With the structured learning method, we can account for the sentence structures and their correspondence between two languages at

[1] This set of labels is quite small and only 16 for the French-English training set with 688,031 sentences.

the same time. Moreover, it keeps the translating structures from source to target.

As we have transformed the tree-to-tree translation problem into a sequence labeling one, all we need to solve is a tagging problem similar to a POS tagging (Daumé III et al. 2006). The input sequence x is word pairs and output y is the group of SHIFT and REDUCE operations. For sequence labeling problem, the standard loss function is Hamming distance, which measures the difference between the true output and the predicting one:

$$HL(y,\hat{y}) = \sum_t \delta(y_t, \hat{y}_t) \quad (1)$$

where δ is 0 if two variables are equal, and 1 otherwise.

5 Decoder

We use a left-to-right beam search decoder to find the best translation given a source sentence. Compared with general phrase-based beam search decoder like Pharaoh (Koehn 2004), this decoder integrates structured information and does not need distortion cost and other costs (e.g. future costs) any more. Therefore, the best translation can be determined by:

$$e^* = \arg\max_e \{ p(f \mid e) p_{lm}(e) \omega^{length(e)} \} \quad (2)$$

where ω is a factor of word length penalty. Similarly, the translation probability $p(f \mid e)$ can be further decomposed into:

$$p(f \mid e) = \prod_i \phi(f_i \mid e_i) \quad (3)$$

and $\phi(f_i \mid e_i)$ represents the probability distribution of word pairs.

Instead of extracting all possible phrases from word alignments, we consider those translation pairs from the nodes of ITG-like trees only. Like Pharaoh, we calculate their probability as a combination of 5 constituents: phrase translation probability (in both directions), lexical translation probability (in both directions) and phrase penalty (default is set at 2.718). The corresponding weight is trained through minimum error rate method (Och 2003). Parameters of this part can be calculated in advance once tree structures are generated and can be stored as phrase translation table.

5.1 Core Algorithm

Another important question is how to preserve sentence structures during decoding. A left-to-right monotonous search procedure is needed. Giving the source sentence, word translation candidates can be determined according to the translation table. Then, several rich features like current and previous source words are extracted based on these translation pairs and source sentence. After that, our structured prediction learning method will be used to predict the output "labels", which produces a bilingual parsing tree. Then, a target output will be generated for the current partial source sentence as soon as bilingual parsing trees are formed. The output of this part therefore contains syntactic information for structure.

For instance, given the current source partial like "f1 f2", we can generate their translation word pair sequences with the translation table, like "f1/e1 f2/e2", "f1/e3 f2/e4" and so on. The corresponding features are then able to be decided for the next predicting process. Once the output predictions (i.e. stack operations) are decided, the bilingual tree structures are formed at the same time. As a consequence, results of these operations are the final translations which we really need.

At each stage of translation, language model parameters can be added to adjust the total costs of translation candidates and make the pruning process reasonable. The whole sentence is then processed by incrementally constructing the translation hypotheses. Lastly, the element in the last beam with the minimum cost is the final translation. In general, the translation process can be described in the following way:

1. Generating all the translation options based on the phrase translation table
2. For each source word in sequence, choose one translation counterpart, generate its word pairs and current word pair sequence for sentence
3. Extract features for structured prediction
4. Using SEARN to predict the "label" sequence and then its parsing tree
5. Generate current target output with recombining and pruning
6. Add the next source word and repeat step 2 until all is finished
7. Output the one with the minimum cost in the last beam.

5.2 Recombining and Pruning

Different translation options can combine to form the same fragment by beam search decoder. Recombining is therefore needed here to reduce the search space. So, only the one with the lowest cost is kept when several fragments are identical. This recombination is a risk-free operation to improve searching efficiency.

Another pruning method used in our system is histogram pruning. Only n-best translations are

allowed for the same source part in each stack (e.g. n=100). In contrast with traditional beam search decoder, we generate our translation candidates from the same input, instead of all allowed word pairs elsewhere. Therefore the pruning is much more reasonable for each beam. There is no relative threshold cut off compared with Pharaoh.

In the end, the complexities for decoding are the main concern of our method. In practice, however, it will not exceed the $O(m * N * Tn)$ (m for sentence length, N for stack size and Tn for allowed translation candidates). This is based on the assumption that our prediction process (tackled by SEARN) is fed with three features (only one former item is associated), which makes it no need of full sentence predictions at each time.

6 Experiment

We validate our method using the corpus from the shared task on NAACL 2006 workshop for statistical machine translation[2]. The difference of our method lies in the framework and different phrase translation table. Experiments are carried on all the three language pairs (French-English, German-English and Spanish-English) and performances are evaluated by the providing test sets. System parameters are adjusted with development data under minimum error rate training.

For SEARN, three features are chosen to use: the current source word, the word before it and the current target word. As we do not know the real target word order before decoding, the corresponding target word's position cannot be used as features. Besides, we filter the features less than 5 times to reduce the training complexities.

The classifier we used in the training process is based on perceptron because of its simplicity and performance. We modified Daumé III's script[3] to fit our method and use the default 5 iterations for each perceptron-based training and 3 itertaions for SEARN.

6.1 Results for different language pairs

The final results of our system, named **Amasis**, and baseline system Pharaoh (Koehn and Monz 2006) for three language pairs are listed in Table 1. The last three lines are the results of Pharaoh with phrase length from 1 to 3. However, the length of

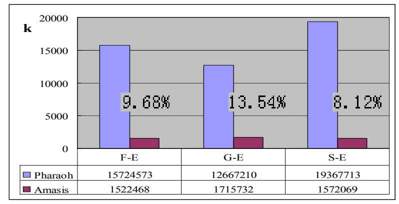


Figure 4: Numbers of translation table

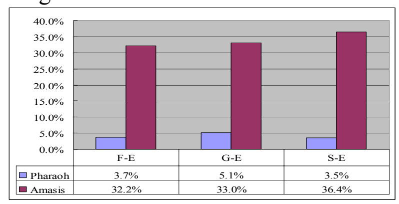


Figure 5: Percent of single word translation pairs (only one word in the source side)

	F-E		G-E		S-E	
	In	Out	In	Out	In	Out
Amasis	27.44	18.41	23.02	15.97	27.51	23.35
$Pharaoh_1$	20.54	14.07	17.53	12.13	23.23	20.24
$Pharaoh_2$	27.71	19.41	23.36	15.77	28.88	25.28
$Pharaoh_3$	30.01	20.77	24.40	16.58	30.58	26.51

Table 1: BLEU scores for different language pairs. In - In-domain test, Out - Out-of-domain test.

phrases for Amasis is determined by ITG-like tree nodes and there is no restriction for it.

Even without producing higher BLEU scores than Pharaoh, our approach is still interesting for the following reasons. First, the number of phrase translation pairs is greatly reduced in our system. The ratio of translation table number in our method (Amasis) to Pharaoh, for French-English is 9.68%, for German-English is 13.54%, for Spanish-English is 8.12% (Figure 4). This means that our method is more efficient at combining words and phrases during translation. The reasons for the different ratio for the three languages are not very clear, maybe are related to the flexibility of word order of source language. Second, we count the single word translation pairs (only one word in the source side) as shown in Figure 5. There are significantly more single word translations in our method. However, the translation quality can be kept at the same level under this circumstance. Third, our current experimental results are produced with only three common features (the corresponding current source and target word and the last source one) without any linguistics information. More useful features are expected to be helpful like POS tags. Finally, the performance can be further improved if we use a more powerful classifier (such as SVM or ME) with more iterations.

[2] http://www.statmt.org/wmt06/shared-task/
[3] http://www.cs.utah.edu/~hal/searn/SimpleSearn.tgz

7 Conclusion

Our method provides a simple and efficient way to solve the word ordering problem partially which is NP-hard (Knight 1999). It is word based except for those indecomposable word sequences under ITGs. However, it can achieve comparable results with phrase-based method (like Pharaoh), while much fewer translation options are used. For the structure prediction process, only 3 common features are preserved and perceptron-based classifiers are chosen for the use of simplicity. We argue that this approach is promising when more features and more powerful classifiers are used as Daumé III et al. (2007) stated.

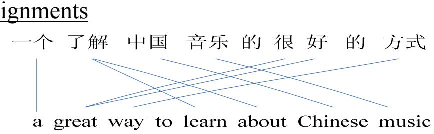
Our contributions lie in the integration of structure prediction for bilingual parsing trees through serial transformations. We reinforce the power of formally syntax-based method by using structured prediction method to obtain tree-to-tree transductions by the transforming from word alignments to ITG-like trees and then to label sequences. Thus, the sentence structures can be better accounted for during translating.

Acknowledgements

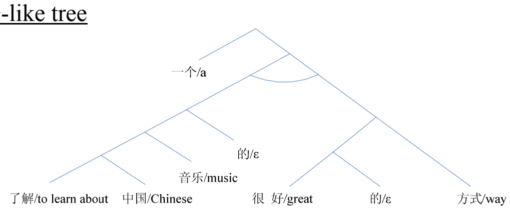
This work is partially supported by National Natural Science Foundation of China under grant #60773027, #60736044 and by "863" Key Projects #2006AA010108. We would like to thank anonymous reviewers for their detailed comments.

Appendix A. A Complete Example in Chinese-English with Our Defined Operations

Word alignments



ITG-like tree



SHIFT-REDUCE label sequence

一个/a	S
了解/to learn about	S
中国/Chinese	S,R+
音乐/music	S,R+
的/ε	S,R+
很 好/great	S
的/ε	S,R+
方式/way	S,R+,R-,R+

Stack status when operations finish

一个 了解 中国 音乐 的 很 好 的 方式
/ a great way to learn about Chinese music

References

A. Burbank, M. Carpuat, et al. 2005. Final Report of the 2005 Language Engineering Workshop on Statistical Machine Translation by Parsing. Johns Hopkins University

D. Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In ACL, pages 263-270.

M. Galley, M. Hopkins, et al. 2004. What's in a translation rule? In HLT-NAACL, Boston, MA.

D. Gildea. 2003. Loosely Tree-Based Alignment for Machine Translation. In ACL, pages 80-87, Sapporo, Japan.

H. Daumé III, J. Langford, et al. 2007. Search-based Structured Prediction. Under review by the Machine Learning Journal. http://pub.hal3.name/daume06searn.pdf.

H. Daumé III, J. Langford, et al. 2006. Searn in Practice. http://pub.hal3.name/daume06searn-practice.pdf.

K. Knight. 1999. Decoding Complexity in Word-Replacement Translation Models. Computational Linguistics 25(4): 607-615.

K. Knight and J. Graehl. 2005. An Overview of Probabilistic Tree Transducers for Natural Language Processing. In CICLing, pages 1-24.

P. Koehn. 2004. Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In Proc. of AMTA, pages 115-124.

P. Koehn and C. Monz. 2006. Manual and Automatic Evaluation of Machine Translation between European Languages. In Proc. on the Workshop on Statistical Machine Translation, pages 102-121, New York City.

P. Koehn, F. J. Och, et al. 2003. Statistical Phrase-Based Translation. In HLT-NAACL, pages 127-133.

P. Liang, A. Bouchard, et al. 2006. An End-to-End Discriminative Approach to Machine Translation. In ACL.

Y. Liu, Q. Liu, et al. 2006. Tree-to-String Alignment Template for Statistical Machine Translation. In ACL.

F. J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In ACL, pages 160-167.

F. J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1): 19-51.

K. Sagae and A. Lavie. 2005. A Classifier-Based Parser with Linear Run-Time Complexity. In IWPT, pages 125-132.

C. Tillmann and T. Zhang. 2005. A Localized Prediction Model for Statistical Machine Translation. In ACL.

C. Tillmann and T. Zhang. 2006. A Discriminative Global Training Algorithm for Statistical MT. in ACL.

J. Turian, B. Wellington, et al. 2006. Scalable Discriminative Learning for Natural Language Parsing and Translation. In Proceedings of NIPS, Vancouver, BC.

T. Watanabe, J. Suzuki, et al. 2007. Online Large-Margin Training for Statistical Machine Translation. In EMNLP.

D. Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. Computational Linguistics 23(3): 377-404.

D. Xiong, Q. Liu, et al. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In ACL, pages 521-528.

K. Yamada and K. Knight. 2001. A Syntax-based Statistical Translation Model. In ACL, pages 523-530.

K. Yamada and K. Knight. 2002. A Decoder for Syntax-based Statistical MT. In ACL, pages 303-310.

Method of Selecting Training Data to Build a Compact and Efficient Translation Model

Keiji Yasuda[†,‡], **Ruiqiang Zhang**[†,‡], **Hirofumi Yamamoto**[†,‡] and **Eiichiro Sumita**[†,‡]
[†]National Institute of Communications Technology
[‡]ATR Spoken Language Translation Research Laboratories
2–2–2, Hikaridai, "Keihanna Science City", Kyoto, 619–0288 Japan
{keiji.yasuda,ruiqiang.zhang}@nict.go.jp
{hirofumi.yamamoto,eiichiro.sumita}@nict.go.jp

Abstract

Target task matched parallel corpora are required for statistical translation model training. However, training corpora sometimes include both target task matched and unmatched sentences. In such a case, training set selection can reduce the size of the translation model. In this paper, we propose a training set selection method for translation model training using linear translation model interpolation and a language model technique. According to the experimental results, the proposed method reduces the translation model size by 50% and improves BLEU score by 1.76% in comparison with a baseline training corpus usage.

1 Introduction

Parallel corpus is one of the most important components in statistical machine translation (SMT), and there are two main factors contributing to its performance. The first is the quality of the parallel corpus, and the second is its quantity.

A parallel corpus that has similar statistical characteristics to the target domain should yield a more efficient translation model. However, domain-mismatched training data might reduce the translation model's performance. A large training corpus obviously produces better quality than a small one. However, increasing the size of the training corpus causes another problem, which is increased computational processing load. This problem not only affects the training of the translation model, but also its applications. The reason for this is that a large amount of training data tends to yield a large translation model and applications then have to deal with this model.

We propose a method of selecting translation pairs as the training set from a training parallel corpus to solve the problem of an expanded translation model with increased training load. This method enables an adequate training set to be selected from a large parallel corpus by using a small in-domain parallel corpus. We can make the translation model compact without degrading performance because this method effectively reduces the size of the set for training the translation model. This compact translation model can outperform a translation model trained on the entire original corpus.

This method is especially effective for domains where it is difficult to enlarge the corpus, such as in spoken language parallel corpora (Kikui et al., 2006). The main approach to recovering from an undersupply of the in-domain corpus has been to use a very large domain-close or out-of-domain parallel corpus for the translation model training (NIST, 2006). In such case, the proposed method effectively reduces the size of the training set and translation model.

Section 2 describes the method of selecting the training set. Section 3 details the experimental results for selecting the training set and actual translation from the International Workshop on Spoken Language Translation 2006 (IWSLT2006). Section 4 compares the results of the proposed method with those of the conventional method. Section 5 concludes the paper.

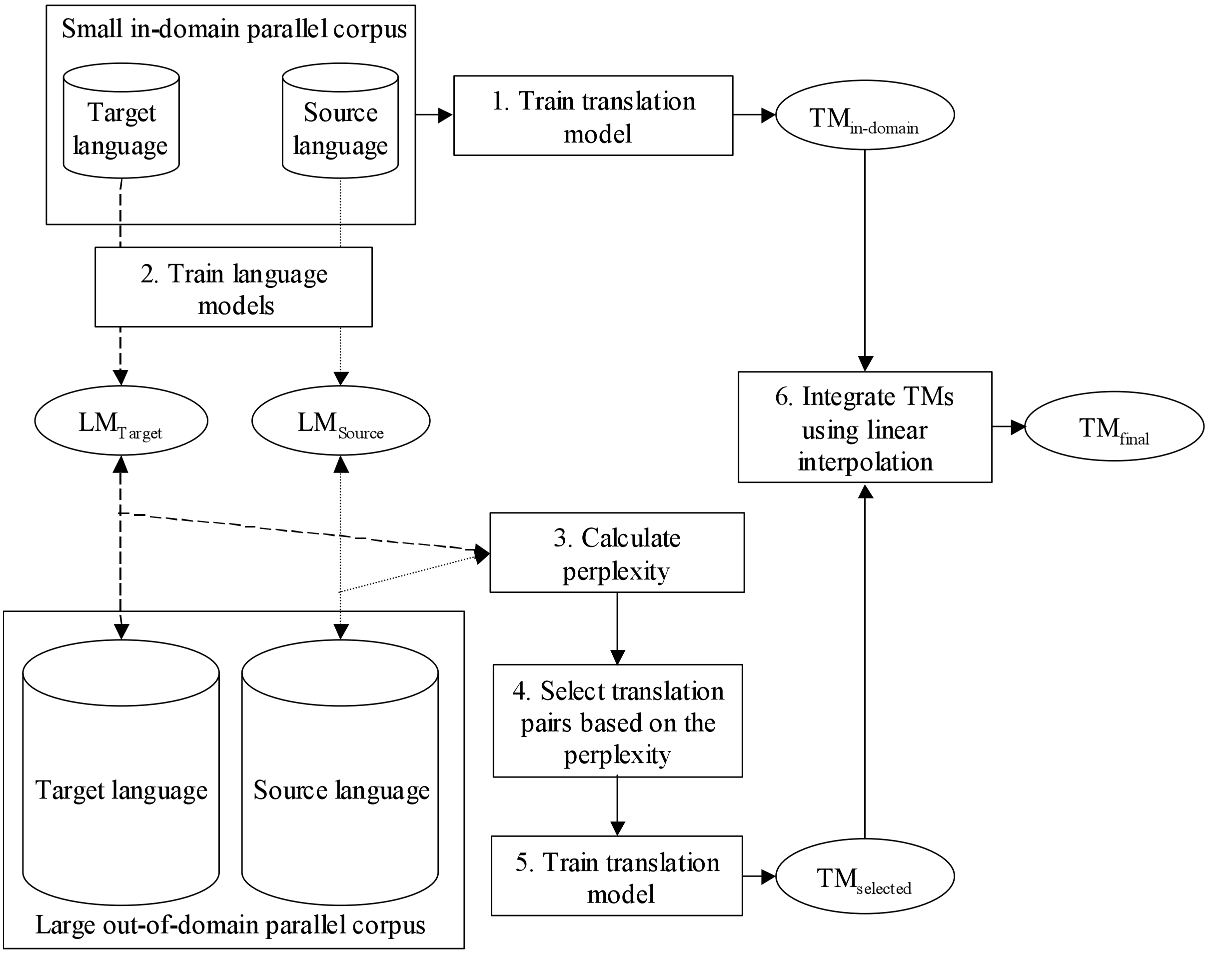


Figure 1: Framework of method.

2 Method

Our method use a small in-domain parallel corpus and a large out-of-domain parallel corpus, and it selects a number of appropriate training translation pairs from the out-of-domain parallel corpus. Figure 1 is a flow diagram of the method. The procedure is as follows:

1. Train a translation model using the in-domain parallel corpus.

2. Train a language model using the source language side or/and target language side of the in-domain corpus.

3. Calculate the word perplexity for each sentence (in source language side or/and target language side) in the out-of-domain corpus by using the following formulas.

$$PP_e = P_e(S_e)^{-\frac{1}{n_e}} \tag{1}$$

where PP_e is the target language side perplexity, and P_e is the probability given by the target side language model. S_e is the target language sentence in the parallel corpus, and n_e is the number of words in the sentence.

We can also calculate the perplexity in the source language (PP_f) in the same way.

$$PP_f = P_f(S_f)^{-\frac{1}{n_f}} \tag{2}$$

If we use perplexities in both languages, we can calculate average perplexity (PP_{e+f}) by using the following formula.

$$PP_{e+f} = (PP_e \times PP_f)^{\frac{1}{2}} \tag{3}$$

Table 1: Size of parallel corpora

	# of sentences		# of words		Explanation
	English	Chinese	English	Chinese	
In-domain parallel corpus	40 K	40 K	320 K	301 K	Basic Travel Expressions Corpus
Out-of-domain parallel corpus	2.5 M	2.5 M	62 M	54 M	LDC corpus (LDC 2002T01, LDC2003T17, LDC2004T07, LDC2004T08, LDC2005T06 and LDC2005T10)

4. Select translation pairs from the out-of-domain parallel corpus. If the perplexity is smaller than the threshold, use translation pairs as the training set. Otherwise, discard the translation pairs.

5. Train a translation model by using the selected translation pairs.

6. Integrate the translation model obtained in 1 and 6 by linear interpolation.

3 Experiments

We carried out statistical machine translation experiments using the translation models obtained with the proposed method.

3.1 Framework of SMT

We employed a log-linear model as a phrase-based statistical machine translation framework. This model expresses the probability of a target-language word sequence (e) of a given source language word sequence (f) given by

$$P(e|f) = \frac{\exp\left(\sum_{i=1}^{M} \lambda_i h_i(e,f)\right)}{\sum_{e'} \exp\left(\sum_{i=1}^{M} \lambda_i h_i(e',f)\right)} \quad (4)$$

where $h_i(e, f)$ is the feature function, λ_i is the feature function's weight, and M is the number of features. We can approximate Eq. 4 by regarding its denominator as constant. The translation results ($\hat{e}$) are then obtained by

$$\hat{e}(f, \lambda_1^M) = \mathrm{argmax}_e \sum_{i=1}^{M} \lambda_i h_i(e,f) \quad (5)$$

3.2 Experimental conditions

3.2.1 Corpus

We used data from the Chinese-to-English translation track of the IWSLT 2006(IWSLT, 2006) for the experiments. The small in-domain parallel corpus was from the IWSLT workshop. This corpus was part of the ATR Bilingual Travel Expression Corpus (ATR-BTEC) (Kikui et al., 2006). The large out-of-domain parallel corpus was from the LDC corpus (LDC, 2007). Details on the data are listed in Table 1. We used the test set of the IWSLT2006 workshop for the evaluation. This test set consisted of 500 Chinese sentences with eight English reference translations per Chinese sentence.

For the statistical machine-translation experiments, we first aligned the bilingual sentences for preprocessing using the Champollion tool (Ma, 2006). We then segmented the Chinese words using Achilles (Zhang et al., 2006). After the segmentation, we removed all punctuation from both English and Chinese corpuses and decapitalized the English corpus. We used the preprocessed data to train the phrase-based translation model by using GIZA++ (Och and Ney, 2003) and the Pharaoh tool kit (Koehn et al., 2003).

3.2.2 Features

We used eight features (Och and Ney, 2003; Koehn et al., 2003) and their weights for the translations.

1. Phrase translation probability from source language to target language (weight = 0.2)

2. Phrase translation probability from target language to source language (weight = 0.2)

3. Lexical weighting probability from source language to target language (weight = 0.2)

4. Lexical weighting probability from source target to language weight = 0.2)

5. Phrase penalty (weight = 0.2)

6. Word penalty (weight = -1.0)
7. Distortion weight (weight = 0.5)
8. Target language model probability (weight = 0.5)

According to a previous study, the minimum error rate training (MERT) (Och, 2003), which is the optimization of feature weights by maximizing the BLEU score on the development set, can improve the performance of a system. However, the range of improvement is not stable because the MERT algorithm uses random numbers while searching for the optimum weights. As previously mentioned, we used fixed weights instead of weights optimized by MERT to remove its unstable effects and simplify the evaluation.

3.2.3 Linear interpolation of translation models

The experiments used four features (Feature # 1 to 4 in 3.2.2) as targets for integration. For each feature, we applied linear interpolation by using the following formula.

$$h(e, f) = \mu_{out} h_{out}(e, f) + (1 - \mu_{out}) h_{in}(e, f) \quad (6)$$

Here, $h_{in}(e, f)$ and $h_{out}(e, f)$ are features trained on the in-domain parallel corpus and out-of-domain corpus, respectively. μ_{out} is the weight for the feature trained on the out-of-domain parallel corpus.

3.2.4 Language model

We used a Good-Turing (Good, 1953) 3-gram language model for data selection.

For the actual translation, we used a modified Kneser-Ney (Chen and Goodman, 1998) 3-gram language model because modified Kneser-Ney smoothing tended to perform better than the Good-Turing language model in this translation task. For training of the language model, only the English side of the in-domain corpus was used. We used the same language model for the entire translation experiment.

3.3 Experimental results

3.3.1 Translation performance

Figure 2 and 3 plot the results of the experiments. The horizontal axis represents the weight for the out-of-domain translation model, and the vertical axis

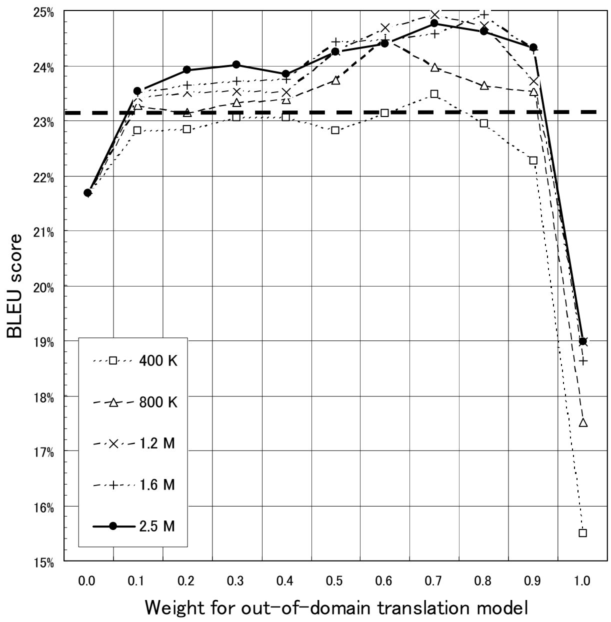


Figure 2: Results of data selection and linear interpolation (BLEU)

represents the automatic metric of translation quality (BLEU score (Papineni et al., 2002) in Fig. 2, and NIST score (NIST, 2002) in Fig. 3). Thick straight broken lines in the figures indicate automatic scores of a baseline system. This base line system was trained on the in-domain and all of the out-of-domain corpus (2.5M sentence pairs). These data were concatenated before training; then one model was trained without linear interpolation. The five symbols in the figures represent the sizes (# of sentence pairs) of the selected parallel corpus. Here, the selection was carried out by using Eq. 1. For automatic evaluation, we used the reference translation with a case unsensitive and no-punctuation setting. Hence, higher automatic scores indicate better translations; the selected corpus size of 1.2M ($\times$) indicates the best translation quality in Fig. 2 at the point where the weight for the out-of-domain translation model is 0.7.

In contrast to Fig. 2, Fig. 3 shows no improvements to the NIST score by using the baseline out-of-domain usage. The optimal weights for each corpus size are different from those in Fig. 2. However, there is no difference in optimal corpus size; i.e., the selected corpus size of 1.2M gives the best NIST score.

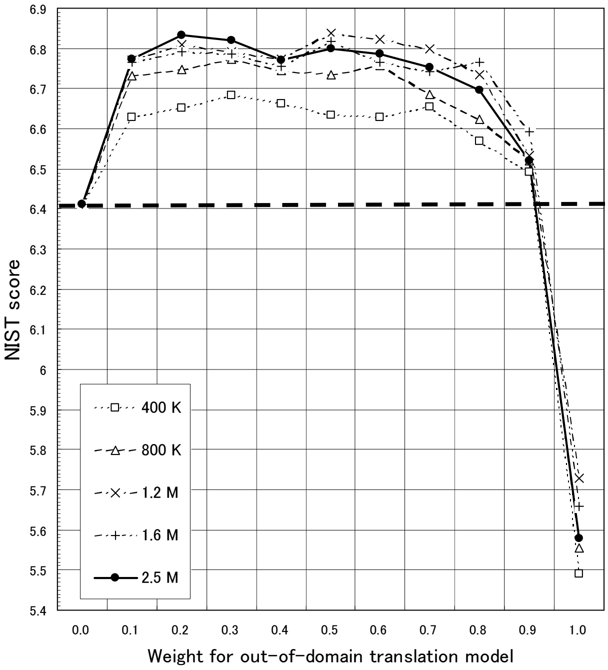


Figure 3: Results of data selection and linear interpolation (BLEU)

Table 2: Size of integrated phrase tables

Corpus size (Sentence pairs)		Size of phrase table (Bytes)
In-domain	Out-of-domain	
40 K	0	14 M
40 K	1.2 M	917 M
40 K	2.5 M	1.8 G

3.3.2 Size of the translation models

Table 2 lists the sizes of the translation models of the baseline and optimum-size training corpus. The size of the phrase table is the uncompressed file size of the phrase table trained by the Pharaoh tool kit. As the table indicates, our method reduced the model sizes by 50%.

This reduction had a positive effect on the computational load of decoding.

3.3.3 Equations for the selection

The experiments described above used only target language side information, i.e., Eq. 1, for the data selection. Here, we compare selection performances of Eqs. 1, 2, and 3. Table 3 shows the results. The first row shows the results of using only the in-domain parallel corpus. The second row shows results of the baseline. The third row shows the results of using linear interpolation without data selection. Comparing the results for the three equations, we see that Eq. 1 gives the best performance. It outperforms not only the baseline but also the results obtained by using all of the (2.5M) out-of-domain data and linear interpolation.

The results of using source language side information (Eq. 2) and information from both language sides (Eq. 3) still showed better performance than the baseline system did.

4 Comparison with conventional method

There are few studies on data selection for translation model training. Most successful and recent study was that of (Lu et al., 2007). They applied the TF*IDF framework to translation model training corpus selection. According to their study, they obtained a 28% translation model size reduction (A 2.41G byte model was reduced to a 1.74G byte model) and 1% BLEU score improvement (BLEU score increased from 23.63% to 24.63%). Although there results are not directly comparable to ours [??] because of the differences in the experimental setting, our method outperforms theirs for both aspects of model size reduction and translation performance improvement (50% model size reduction and 1.76% BLEU score improvement).

5 Conclusions

We proposed a method of selecting training sets for training translation models that dramatically reduces the sizes of the training set and translation models.

We carried out experiments using data from the Chinese-to-English translation track of the IWSLT evaluation campaign. The experimental results indicated that our method reduced the size of the training set by 48%. The obtained translation models were half the size of the baseline.

The proposed method also had good translation performance. Our experimental results demonstrated that an SMT system with a half-size translation model obtained with our method improved the BLEU score by 1.76%. (Linear interpolation improved BLEU score by 1.61% and data selection improved BLEU score by an additional 0.15%.)

Table 3: Results of data selection by using Eqs. 1, 2, and 3

Corpus size (Sentence pairs)		Selection method	Optimal weight for out-of-domain model	BLEU score
In-domain	Out-of-domain			
40 K	0	N/A	N/A	21.68%
40 K	2.5 M	N/A	N/A	23.16%
40 K	2.5 M	N/A	0.7	24.77%
40 K	1.2 M	Eq. 1	0.7	24.92%
40 K	1.2 M	Eq. 2	0.8	24.76%
40 K	1.2 M	Eq. 3	0.6	24.56%

We also compared the selections using source language side information, target language side information and information from both language sides. The experimental results show that target language side information gives the best performance in the experimental setting. However, there are no large differences among the different selection results. The results are encouraging because they show that the in-domain mono-lingual corpus is sufficient to select training data from the out-of-domain parallel corpus.

References

S. F. Chen and J. Goodman. 1998. An empirical study of smoothing techniques for language modeling. In *Technical report TR-10-98, Center for Research in Computing Technology (Harvard University)*.

I. J Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3):237–264.

IWSLT. 2006. IWSLT: International Workshop on Spoken Language Translation. http://www.slc.atr.jp/IWSLT2006/.

G. Kikui, S. Yamamoto, T. Takezawa, and E. Sumita. 2006. Comparative study on corpora for speech translation. In *IEEE Transactions on Audio, Speech and Language Processing*, volume 14(5), pages 1674–1682.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. *Proc. of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 127–133.

LDC. 2007. Linguistic Data Consortium. http://www.ldc.upenn.edu/.

Yajuan Lu, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 343–350.

X Ma. 2006. Champollion: A Robust Parallel Text Sentence Aligner. In *Proc. of international conference on Language Resources and Evaluation (LREC)*, pages 489–492.

NIST. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurence Statistics. http://www.nist.gov/speech/tests/mt/mt2001/resource/.

NIST. 2006. The 2006 NIST Machine Translation Evaluation Plan (MT06). http://www.nist.gov/speech/tests/mt/doc/mt06_evalplan.v3.pdf.

F. J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

F. J. Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

R. Zhang, G. Kikui, and E. Sumita. 2006. Subword-based Tagging by Conditional Random Fields for Chinese Word Segmentation. *Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Short Paper:193–196.

Large and Diverse Language Models for Statistical Machine Translation

Holger Schwenk*
LIMSI - CNRS
France
schwenk@limsi.fr

Philipp Koehn
School of Informatics
University of Edinburgh
Scotland
pkoehn@inf.ed.ac.uk

Abstract

This paper presents methods to combine large language models trained from diverse text sources and applies them to a state-of-art French–English and Arabic–English machine translation system. We show gains of over 2 BLEU points over a strong baseline by using continuous space language models in re-ranking.

1 Introduction

Often more data is better data, and so it should come as no surprise that recently statistical machine translation (SMT) systems have been improved by the use of large language models (LM). However, training data for LMs often comes from diverse sources, some of them are quite different from the target domain of the MT application. Hence, we need to weight and combine these corpora appropriately. In addition, the vast amount of training data available for LM purposes and the desire to use high-order n-grams quickly exceeds the conventional computing resources that are typically available. If we are not able to accommodate large LMs integrated into the decoder, using them in re-ranking is an option.

In this paper, we present and compare methods to build LMs from diverse training corpora. We also show that complex LMs can be used in re-ranking to improve performance given a strong baseline. In particular, we use high-order n-grams continuous space LMs to obtain MT of the well-known NIST 2006 test set that compares very favorably with the results reported in the official evaluation.

*new address: LIUM, University du Maine, France, Holger.Schwenk@lium.univ-lemans.fr

2 Related Work

The utility of ever increasingly large LMs for MT has been recognized in recent years. The effect of doubling LM size has been powerfully demonstrated by Google's submissions to the NIST evaluation campaigns. The use of billions of words of LM training data has become standard in large-scale SMT systems, and even trillion word LMs have been demonstrated. Since lookup of LM scores is one of the fundamental functions in SMT decoding, efficient storage and access of the model becomes increasingly difficult.

A recent trend is to store the LM in a distributed cluster of machines, which are queried via network requests (Brants et al., 2007; Emami et al., 2007). It is easier, however, to use such large LMs in re-ranking (Zhang et al., 2006). Since the use of clusters of machines is not always practical (or affordable) for SMT applications, an alternative strategy is to find more efficient ways to store the LM in the working memory of a single machine, for instance by using efficient prefix trees and fewer bits to store the LM probability (Federico and Bertoldi, 2006). Also the use of lossy data structures based on Bloom filters has been demonstrated to be effective for LMs (Talbot and Osborne, 2007a; Talbot and Osborne, 2007b). This allows the use of much larger LMs, but increases the risk of errors.

3 Combination of Language Models

LM training data may be any text in the output language. Typically, however, we are interested in building a MT system for a particular domain. If text resources come from a diversity of domains, some may be more suitable than others. A common strat-

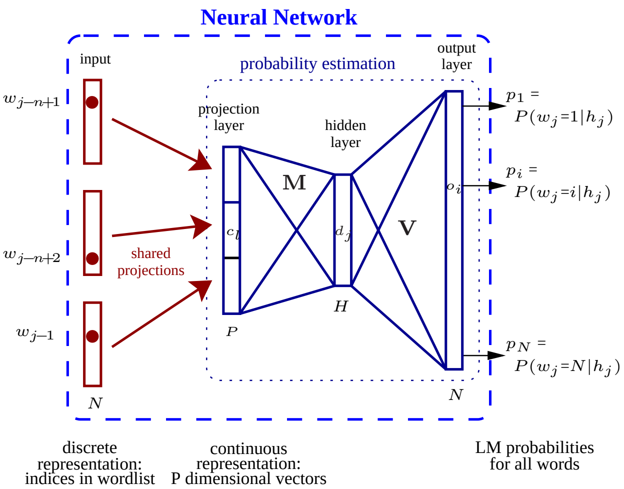


Figure 1: Architecture of the continuous space LM.

egy is to divide up the LM training texts into smaller parts, train a LM for each of them and combine these in the SMT system. Two strategies may be employed to combine LMs: One is the use of interpolation. LMs are combined into one by weighting each based on their relevance to the focus domain. The weighting is carried out by optimizing perplexity of a representative tuning set that is taken from the domain. Standard LM toolkits like SRILM (Stolcke, 2002) provide tools to estimate optimal weights using the EM algorithm.

The second strategy exploits the log-linear model that is the basis of modern SMT systems. In this framework, a linear combination of feature functions is used, which include the log of the LM probability. It is straight-forward to use multiple LMs in this framework and treat each as a feature function in the log-linear model. Combining several LMs in the log domain corresponds to multiplying the corresponding probabilities. Strictly speaking, this supposes an independence assumption that is rarely satisfied in practice. The combination coefficients are optimized on a criterion directly related to the translation performance, for instance the BLEU score.

In summary, these strategies differ in two points: linear versus log-linear combination, and optimizing perplexity versus optimizing BLEU scores.

4 Continuous Space Language Models

This LM approach is based a *continuous representation* of the words (Bengio et al., 2003). The basic idea is to convert the word indices to a continuous representation and to use a probability estimator operating in this space. Since the resulting distributions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. This approach was successfully applied to language modeling in small (Schwenk et al., 2006) an medium-sized phrase-based SMT systems (Déchelotte et al., 2007).

The architecture of the continuous space language model (CSLM) is shown in Figure 1. A standard fully-connected multi-layer perceptron is used. The inputs to the neural network are the indices of the $n{-}1$ previous words in the vocabulary $h_j{=}w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}$ and the outputs are the posterior probabilities of *all* words of the vocabulary:

$$P(w_j = i|h_j) \qquad \forall i \in [1, N] \tag{1}$$

where N is the size of the vocabulary. The input uses the so-called 1-of-n coding, i.e., the ith word of the vocabulary is coded by setting the ith element of the vector to 1 and all the other elements to 0. The ith line of the $N \times P$ dimensional projection matrix corresponds to the continuous representation of the ith word.[1] Let us denote c_l these projections, d_j the hidden layer activities, o_i the outputs, p_i their softmax normalization, and m_{jl}, b_j, v_{ij} and k_i the hidden and output layer weights and the corresponding biases. Using these notations, the neural network performs the following operations:

$$d_j = \tanh\left(\sum_l m_{jl}\, c_l + b_j\right) \tag{2}$$

$$o_i = \sum_j v_{ij}\, d_j + k_i \tag{3}$$

$$p_i = e^{o_i} \Big/ \sum_{r=1}^{N} e^{o_r} \tag{4}$$

The value of the output neuron p_i corresponds directly to the probability $P(w_j\!=\!i|h_j)$.

Training is performed with the standard back-propagation algorithm minimizing the following error function:

$$E = \sum_{i=1}^{N} t_i \log p_i + \beta \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2\right) \tag{5}$$

[1]Typical values are $P = 200 \ldots 300$

where t_i denotes the desired output. The parameter β has to be determined experimentally. Training is done using a resampling algorithm (Schwenk, 2007). It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities. Therefore, the neural network directly minimizes the perplexity on the training data. Note also that the gradient is back-propagated through the projection-layer, which means that the neural network learns the projection of the words that is best for the probability estimation task.

In general, the complexity to calculate one probability is dominated by the output layer dimension since the size of the vocabulary (here N=273k) is usually much larger than the dimension of the hidden layer (here H=500). Therefore, the CSLM is only used when the to be predicted word falls into the 8k most frequent ones. While this substantially decreases the dimension of the output layer, it still covers more than 90% of the LM requests. The other requests are obtained from a standard back-off LM. Note that the full vocabulary is still used for the words in the context (input of the neural network).

The incorporation of the CSLM into the SMT system is done by using n-best lists. In all our experiments, the LM probabilities provided by the CSLM are added as an additional feature function. It is also possible to use only one feature function for the modeling of the target language (interpolation between the back-off and the CSLM), but this would need more memory since the huge back-off LM must be loaded during n-best list rescoring.

We did not try to use the CSLM directly during decoding since this would result in increased decoding times. Calculating a LM probability with a back-off model corresponds basically to a table look-up, while a forward pass through the neural network is necessary for the CSLM. Very efficient optimizations are possible, in particular when n-grams with the same context can be grouped together, but a re-organization of the decoder may be necessary.

5 Language Models in Decoding and Re-Ranking

LM lookups are one of the most time-consuming steps in the decoding process, which makes time-efficient implementations essential. Consequently, the LMs have to be held in the working memory of the machine, since disk lookups are simply too slow. Filtering LMs to the n-grams which are needed for the decoding a particular sentence may be an option, but is useful only to a degree. Since the order of output words is unknown before decoding, all n-grams that contain any of output words that may be generated during decoding need to be preserved.

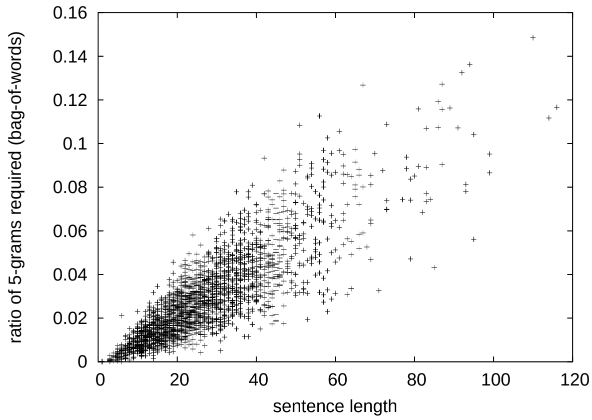


Figure 2: Ratio of 5-grams required to translate one sentence. The graph plots the ratio against sentence length. For a 40-word sentence, typically 5% of the LM is needed (numbers from German–English model trained on Europarl).

See Figure 2 for an illustration that highlights what ratio of the LM is needed to translate a single sentence. The ratio increases roughly linear with sentence length. For a typical 30-word sentence, about 4% of the LM 5-grams may be potentially generated during decoding. For large 100-word sentences, the ratio is about 15%.[2] These numbers suggest that we may be able to use 5–10 times larger LMs, if we filter the LM prior to the decoding of each sentence. SMT decoders such as Moses (Koehn et al., 2007) may store the translation model in an efficient on-disk data structure (Zens and Ney, 2007), leaving almost the entire working memory for LM storage. However, this means for 32-bit machines a limit of 3 GB for the LM.

On the other hand, we can limit the use of very large LMs to a re-ranking stage. In two-pass de-

[2]The numbers were obtained using a 5-gram LM trained on the English side of the Europarl corpus (Koehn, 2005), a German–English translation model trained on Europarl, and the WMT 2006 test set (Koehn and Monz, 2006).

	French	English
News Commentary	1.2M	1.0M
Europarl	37.5M	33.8M

Table 1: Combination of a small in-domain (News Commentary) and large out-of-domain (Europarl) training corpus (number of words).

coding, the initial decoder produces an n-best list of translation candidates (say, n=1000), and a second pass exploits additional features, for instance very large LMs. Since the order of English words is fixed, the number of different n-grams that need to be looked up is dramatically reduced. However, since the n-best list is only the tip of the iceberg of possible translations, we may miss the translation that we would have found with a LM integrated into the decoding process.

6 Experiments

In our experiments we are looking for answers to the open questions on the use of LMs for SMT: Do perplexity and BLEU score performance correlate when interpolating LMs? Should LMs be combined by interpolation or be used as separate feature functions in the log-linear machine translation model? Is the use of LMs in re-ranking sufficient to increase machine translation performance?

6.1 Interpolation

In the WMT 2007 shared task evaluation campaign (Callison-Burch et al., 2007) domain adaptation was a special challenge. Two training corpora were provided: a small in-domain corpus (News Commentary) and the about 30 times bigger out-of-domain Europarl corpus (see Table 1). One method for domain adaptation is to bias the LM towards the in-domain data. We train two LMs and interpolate them to optimize performance on in-domain data. In our experiments, the translation model is first trained on the combined corpus without weighting. We use the Moses decoder (Koehn et al., 2007) with default settings. The 5-gram LM was trained using the SRILM toolkit. We only run minimum error rate training once, using the in-domain LM. Using different LMs for tuning may change our findings reported here.

When interpolating the LMs, different weights may be given to the out-of-domain versus the in-domain LM. One way to tune the weight is to optimize perplexity on a development set (nc-dev2007). We examine values between 0 and 1, the EM procedure gives the lowest perplexity of 193.9 at a value of 0.42. Does this setting correspond with good BLEU scores on the development and test set (nc-devtest2007) ? See Figure 3 for a comparison. The BLEU score on the development data is 28.55 when the interpolation coefficient is used that was obtained by optimizing the perplexity. A slightly better value of 28.78 good be obtained when using an interpolation coefficient of 0.15. The test data seems to be closer to the out-of-domain Europarl corpus since the best BLEU scores would be obtained for smaller values of the interpolation coefficient.

The second question we raised was: Is interpolation of LMs preferable to the use of multiple LMs

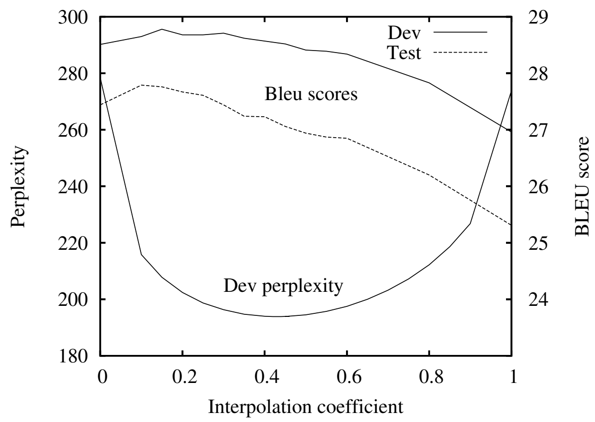


Figure 3: Different weight given to the out-of-domain data and effect on perplexity of a development set (nc-dev2007) and on the BLEU score of the test set (nc-devtest2007).

TM	LM	BLEU (test)
combined	2 features	27.30
combined	interpolated 0.42	27.23
2 features	2 features	27.64
2 features	interpolated 0.42	27.63

Table 2: Combination of the translation models (TM) by simple concatenation of the training data vs. use of two feature functions, and combination of the LM (LM) by interpolation or the use of two feature functions.

as separate feature functions. See Table 2 for numbers in the same experimental setting for two different comparisons. First, we compare the performance of the interpolated LM with the use of two feature functions. The resulting BLEU scores are very similar (27.23 vs. 27.30). In a second experiment, we build two translation models, one for each corpus, and use separate feature functions for them. This gives a slightly better performance, but again it gives almost identical results for the use of interpolated LMs vs. two LMs as separate feature functions (27.63 vs. 27.64).

These experiments suggest that interpolated LMs give similar performance to the use of multiple LMs. In terms of memory efficiency, this is good news, since an interpolated LM uses less memory.

6.2 Re-Ranking

Let us now turn our attention to the use of very large LMs in decoding and re-ranking. The largest freely available training sets for MT are the corpora provided by the LDC for the NIST and GALE evaluation campaigns for Arabic–English and Chinese–English. In this paper, we concentrate on the first language pair. Our starting point is a system using Moses trained on a training corpus of about 200 million words that was made available through the GALE program. Training such a large system pushes the limits of the freely available standard tools.

For instance, GIZA++, the standard tool for word alignment keeps a word translation table in memory. The only way to get it to process the 200 million word parallel corpus is to stem all words to their first five letters (hence reducing vocabulary size). Still, GIZA++ training takes more than a week of compute time on our 3 GHz machines. Training uses default settings of Moses. Tuning is carried out using the 2004 NIST evaluation set. The resulting system is competitive with the state of the art. The best

Corpus	Words
Parallel training data (train)	216M
AFP part of Gigaword (afp)	390M
Xinhua part of Gigaword (xin)	228M
Full Gigaword (giga)	2,894M

Table 3: Size of the training corpora for LMs in number of words (including punctuation)

	Px	**Bleu score**	
Decode LM	**eval04**	**eval04**	**eval06**
3-gram train+xin+afp	86.9	50.57	43.69
3-gram train+giga	85.9	50.53	43.99
4-gram train+xin+afp	74.9	50.99	43.90
Reranking with continuous space LM:			
5-gram train+xin+afp	62.5	52.88	46.02
6-gram train+xin+afp	60.9	53.25	45.96
7-gram train+xin+afp	60.5	52.95	45.96

Table 4: Improving MT performance with larger LMs trained on more training data and using higher order of n-grams (Px denotes perplexity).

performance we obtained is a BLEU score of 46.02 (case insensitive) on the most recent eval06 test set. This compares favorably to the best score of 42.81 (case sensitive), obtained in 2006 by Google. Case-sensitive scoring would drop our score by about 2-3 BLEU points.

To assess the utility of re-ranking with large LMs, we carried out a number of experiments, summarized in Table 4. We used the English side of the parallel training corpus and the Gigaword corpus distributed by the LDC for language modeling. See Table 3 for the size of these corpora. While this puts us into the moderate billion word range of large LMs, it nevertheless stresses our resources to the limit. The largest LMs that we are able to support within 3 GB of memory are a 3-gram model trained on all the data, or a 4-gram model trained only on train+afp+xin. On disk, these models take up 1.7 GB compressed (gzip) in the standard ARPA format. All these LMs are interpolated by optimizing perplexity on the tuning set (eval04).

The baseline result is a BLEU score of 43.69 using a 3-gram trained on train+afp+xin. This can be slightly improved by using either a 3-gram trained on all data (BLEU score of 43.99) or by using a 4-gram trained on train+afp+xin (BLEU score of 43.90). We were not able to use a 4-gram trained on all data during the search. Such a model would take more than 6GB on disk. An option would be to train the model on all the data and to prune or quantize it in order to fit in the available memory. This may give better results than limiting the training data.

Next, we examine if we can get significantly better performance using different LMs in re-ranking.

To this end, we train continuous space 5-gram to 7-gram LMs and re-rank a 1000-best list (without duplicate translations) provided by the decoder using the 4-gram LM. The CSLM was trained on the same data as the back-off LMs. It yields an improvement in perplexity of about 17% relative.

With various higher order n-grams models, we obtain significant gains, up to just over 46 BLEU on the 2006 NIST evaluation set. A gain of over 2 BLEU points underscores the potential for re-ranking with large LM, even when the baseline LM was already trained on a large corpus. Note also the good generalization behavior of this approach : the gain obtained on the test data matches or exceeds in most cases the improvements obtained on the development data. The CSLM is also very memory efficient since it uses a distributed representation that does not increase with the size of training material used. Overall, about 1GB of main memory is used.

7 Discussion

In this paper we examined a number of issues regarding the role of LMs in large-scale SMT systems. We compared methods to combine training data from diverse corpora and showed that interpolation of LMs by optimizing perplexity yields similar results to combining them as feature functions in the log-linear model.

We applied for the first time continuous space LMs to the large-scale Arabic–English NIST evaluation task. We obtained large improvements (over 2 BLEU points) over a strong baseline, thus validating both continuous space LMs and re-ranking as a method to exploit large LMs.

Acknowledgments

This work has been partially funded by the French Government under the project INSTAR (ANR JCJC06 143038) and the DARPA Gale program, Contrat No. HR0011-06-C-0022 and the EuroMatrix funded by the European Commission (6th Framework Programme).

References

Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JMLR*, 3(2):1137–1155.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *EMNLP*, pages 858–867.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (Meta-) evaluation of machine translation. In *Second Workshop on SMT*, pages 136–158.

Daniel Déchelotte, Holger Schwenk, Hlne Bonneau-Maynard, Alexandre Allauzen, and Gilles Adda. 2007. A state-of-the-art statistical machine translation system based on Moses. In *MT Summit*.

Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. 2007. Large-scale distributed language modeling. In *ICASSP*.

Marcello Federico and Nicola Bertoldi. 2006. How many bits are needed to store probabilities for phrase-based translation? In *First Workshop on SMT*, pages 94–101.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *First Workshop on SMT*, pages 102–121.

Philipp Koehn et al. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL Demo and Poster Sessions*, pages 177–180, June.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*.

Holger Schwenk, Marta R. Costa-jussà, and José A. R. Fonollosa. 2006. Continuous space language models for the IWSLT 2006 task. In *IWSLT*, pages 166–173.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21:492–518.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*, pages II: 901–904.

David Talbot and Miles Osborne. 2007a. Randomised language modelling for statistical machine translation. In *ACL*, pages 512–519.

David Talbot and Miles Osborne. 2007b. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *EMNLP*, pages 468–476.

Richard Zens and Hermann Ney. 2007. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *NACL*, pages 492–499.

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. 2006. Distributed language modeling for n-best list re-ranking. In *EMNLP*, pages 216–223.

A linguistic and navigational knowledge approach to text navigation

Javier Couto
Facultad de Ingeniería - UdelaR
Julio Herrera y Reissig 565
11300 – Montevideo – Uruguay
jcouto@fing.edu.uy

Jean-Luc Minel
MoDyCO, UMR 7114 CNRS – Univ. Paris X
200, avenue de la République
92 001 – Nanterre – France
jminel@u-paris10.fr

Abstract

We present an approach to text navigation conceived as a cognitive process exploiting linguistic information present in texts. We claim that the navigational knowledge involved in this process can be modeled in a declarative way with the *Sextant* language. Since *Sextant* refers exhaustively to specific linguistic phenomena, we have defined a customized text representation. These different components are implemented in the text navigation system *NaviTexte*. Two applications of *NaviTexte* are described.

1 Introduction

Text navigation has several interpretations. Usually, this term refers to hypertext systems, which offer the possibility to activate hyperlinks, moving the reading point from a text unit (*source*) to another one (*target*), this change being intra or intertextual. From our point of view, this conception presents some limitations. First, the hyperlink activation is not assisted. In other words, imprecise, poor or no information is provided to the reader before s/he activates the link. Second, the reader does not know where the movement will be carried out in the text (before or after the reading point or outside the text), which generates the "lost in hyperspace" problem (Edwards and Hardman 1989). Finally, hyperlinks are embedded in the hypertext. Therefore, there is no clearly distinction between text constituents and navigation knowledge. In addition, by not explicitly modeling this knowledge, it is not reusable.

Different solutions have been proposed to address the problems mentioned. Some researchers (Danielson, 2002) have tried to mitigate the lost in hyperspace problem offering global maps where the reading point is clearly identified. Adaptive hypertext (Mathe and Chen, 1994; Brusilovsky, 1996) relying on user model, proposes to modify the way the text is shown on the screen. Dynamic hypertext (Bodner and Chignell, 1999) computes the value of hyperlinks using several criteria such as text similarity or predefined relations. In this approach, a hyperlink is defined as a query returning a text node.

In some way, our conception of text navigation is related to the notion of computed query, but rather than taking into account criteria depending on the reader, the target is computed by exploiting linguistic information in texts. Moreover, the queries are not placed in texts but they are encapsulated as knowledge by a specific language (Sextant), which allows isolating the navigational knowledge to create knowledge bases. Both texts and queries (navigational knowledge) are interpreted by NaviTexte, which manages the interactions with a reader.

The remainder of this paper is organized as follows. In the next section, we discuss our approach to text navigation. The third section describes a navigational knowledge modeling language called Sextant. The fourth section details the text navigation system NaviTexte. The fifth section describes two applications of NaviTexte. Then we address the evaluation aspects of our approach. At last, conclusions are presented.

2 Defining text navigation

Our conception of text navigation lies in the hypothesis that navigating through texts is the expression of a cognitive process related to specific knowledge (Minel, 2003; Couto and Minel, 2006). More precisely: we claim that a reader moves through texts applying some knowledge to exploit linguistic information present in texts (*e.g.* discursive markers). Moreover, we claim that this

knowledge may be articulated in a declarative way (*cf.* Sec. 3) relying on information in texts coded, on the one hand, by its structure, and, on the other hand, by specific annotations.

The main difference between classic hypertext and our conception of navigation lies on the status of texts and on the definition of navigational knowledge. In the case of hypertext, the visualization of a text is unique and navigational knowledge is encoded (embedded) in the text. In our approach, there are several ways to visualize a text (Couto, 2006), each way called *text view*, and for each view, different navigational knowledge may be defined. As a consequence, the navigation is not guided by the author, compared to hypertext navigation where s/he determines the links, but it is the result of an interpretation process made by the reader, relying on text structure and annotations.

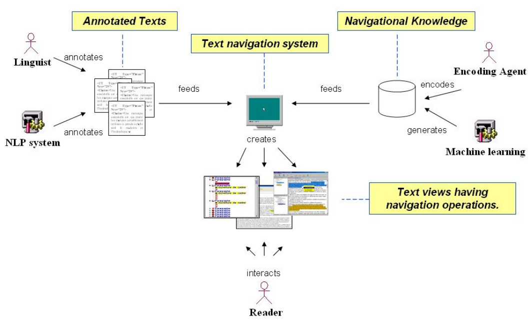


Figure 1. Elements of text navigation.

Our conception of navigation (*cf.* Fig.1) relies on four elements: i) a text representation allowing linguistic specific phenomena and annotations (see Couto, 2006); ii) a language to model navigational knowledge (*cf.* Sec. 3); iii) an agent (an individual or a software) able to encode such knowledge; iv) a system, called NaviTexte, to interpret and apply knowledge to a specific text (*cf.* Sec. 4).

3 Modeling navigational knowledge: the Sextant language

To allow the unambiguous isolation of navigational knowledge we need a formal modeling language. We want to model the knowledge applied by a reader to move through texts, claiming that this knowledge exploits linguistic information present in texts. We do not say that this is the only way a reader may move through texts (*e.g.* strolling courses proposed by Géry (2002) is a counterexample), but we say that this is the kind of way than we are going to model. Different views of a text and the fact that each view contains specific indications of the possible reading courses constitute the heart of the language.

3.1 Knowledge modules ant text views

A text view may be a *full* view or a *partial* view focused in some specific phenomena present in the text (for example a view of all discourse referents). The constituent elements of a view are formalized in a *view description*, which contains the type of view, its parameters, the creation constraints (*i.e.* conditions to verify by the TU of the view) and the navigation operations (see next section). At present, four types of view have been defined: *plain-text*, *tree*, *graph* and *temporality*. The three firsts types are described in (Couto, 2006). The last one graphically represents temporality in texts and a complete description is in (Battistelli *et al.*, 2006).

Several view descriptions may be gathered by the encoder in an entity called *navigational knowledge module*. The creation of a view may be conceptualized as the application of a view description to a specific text. Thus, the application of a module implies the creation of a set of text views.

3.2 The navigation operation

The notion of computed query mentioned in section 1 is formalized in Sextant as a *navigation operation*, which links a *source TU* to a *target TU*. In classic hypertext systems one must explicitly connect the specific source to the specific target. For example, if a reader wants, for all definitions in a scientific paper, to move from one to the following one, several hyperlinks must be defined. In our approach we specify the source and the target using conditions. As a result, we can abstract, for example, the navigational knowledge that states "go from one definition to the following one", being "definition" one of the TU annotations.

We can specify several conditions for the source and the target. We say that a navigation operation is *available* for a TU if this TU verifies the source conditions. A text navigation system should find the TU that verifies the target conditions. As several TU in the text may verify them, we need a way of disambiguation. This is done by the *orientation* parameter, which specifies the direction of the target search by using one of these options: *first*, *last*, *forward(i)*, *backward(i)*. *First* and *last* indicate that the search of the target is absolute: the TU to select

will be the first (respectively the last) TU that verify the conditions. *Forward(i)* and *backward(i)* indicate that the search is carried out relatively to the source (before or after) and indexed by the integer *i*. For example, "forward(3)" is interpreted as the third TU, after the source, of all the TU verifying the target conditions.

3.3 The conditions language

The conditions language is an important component of Sextant ant it is composed by *basic conditions*, *TU elements existence conditions*, *hierarchical conditions* and *non-hierarchical conditions.*

Basic conditions concern TU's attributes and annotations. For this kind of condition we use a notation close to the pattern notion. We define an operator called TU, having five operands that correspond to the following properties: *type*, *number*, *level*, *annotations* and *string*. With the three first operands and the fifth one, we denote constraints of equality, inequality, order, prefix, suffix and substring occurrence. The fourth operand is used to indicate the existence or non-existence of annotations, whether it is an annotation name, a value or a name-value pair.

For *TU elements existence conditions*, we define operators without operands to verify if a TU has *annotations*, *string*, *title*, *parent* and *children*.

For conditions dealing with *hierarchical relationship* between different TU, a set of unary operators have been defined, taking a basic condition as an argument. For example, the *isAscendant* operator verifies if a TU is the ascendant of another TU specified by a basic condition. The other operators are: *isParent*, *isChild*, *isSibling*, *isDescendant*, *hasInTitle*, *isInTitle*. We would like to draw attention to the fact that these operators allow to move through the hierarchy of TU from a starting TU (Couto, 2006).

Non-hierarchical conditions concern constructed units' attributes and annotations as well as TU constitution.

All conditions may be combined using the classic logic operators OR, AND and NOT. Figure 2 presents an example of a language expression that formulates the following condition: TU of type "NP", having an annotation of name "discourse referent", for which it exists, among its descendants, a TU of type "paragraph" not having an annotation of name "semantic label" whose value is "conclusion".

TU(type = NP, *,*,{(discourse referent, *)},*) **AND**
isDescendant(**TU**(type = paragraphe,*,*, {¬∃(semantic label, conclusion)},*))

Figure 2. Conditions language example.

This condition means: noun phrases being a discourse referent that does not occur in a concluding paragraph.

4 NaviTexte: a text navigation system

Several adaptive navigation systems have been proposed (Benyon and Murray, 1993; Kaplan *et al.*, 1993; Boyle and Encarnacion, 1994; Brusilovsky and Pesin, 1994; Brusilovsky *et al.*, 1996). While they are goal specific (learning, tutoring, reading, etc.), NaviTexte (Couto, 2006) is a generic text navigation system implementing our approach. This means that, depending on texts and knowledge modules, NaviTexte may be used, for example, as a learning, tutoring or reading system. Another important difference is that NaviTexte gives the user the liberty to navigate through the text following its own interests (the system propose - the reader chooses), while the mentioned systems try to maintain a user stuck to a given route (the user chooses - the system propose) (Höök and Svensson, 1999).

NaviTexte consists of sub-systems dealing with: *text representation*, *navigational knowledge*, *visual representation* and *user interaction*. The first one builds a text representation in memory from a text annotated manually or by dedicated software (Cunningham *et al.*, 2002; Bilhaut *et al.*, 2003). The second sub-system loads and compiles the knowledge modules. The result of this compilation is a graph of potential navigation courses that in practice is calculated as needed and stored in optimization data structures. The third sub-system calculates and displays different text views and the fourth one manages the user interaction.

The reader has the possibility to load and unload several texts and knowledge modules in the same work session. A complete description of NaviTexte may be found in (Couto, 2006).

5 Applications of NaviTexte

Building an application with NaviTexte requires a set of texts and navigational knowledge modules. Both text representation and Sextant language have XML implementations with dedicated editors to

use in case of a manual text annotation and a human knowledge encoder, respectively (*cf.* Fig.1).

So far four applications have been developed: *alternative automatic summarization* (Couto and Minel, 2006), *the NaviLire project* (Couto *et al.*, 2005; Lunquist *et al.*, 2006), *re-reading Madame Bovary* (Couto and Minel, 2006) and *temporality in texts* (Battistelli *et al.*, 2006). We present two of them to illustrate NaviTexte's potential.

5.1 NaviLire: a text linguistics application

For the past thirty years, text linguistic researchers have worked on describing linguistic markers of textual coherence in order to bring out principles of text structuring (Lundquist, 1980). A set of concepts and models of textual interpretation has been worked out, including for example, anaphora, connectors, mental spaces, etc. In particular, these studies have shown that even for languages apparently close like French and Danish, texts are not organized in the same way (Lundquist, 2005). Consequently, text linguistics has important implications in foreign language teaching, especially from a contrastive point of view, when language pairs are analyzed through texts used in authentic communication situations. It seems that the teaching of text linguistics contributes to sharpen the attention of students towards the building of well-formed texts and to stimulate their own text production. Therefore, a tool that allows the student to perceive text units that contribute to and maintain text coherence and to navigate between them, can be supposed to be an important didactic tool for teaching reading of foreign language texts, as well as producing written texts in the foreign language.

In the reading process, the student has to deal with two basic types of cognitive problems. First, s/he has to identify discourse referents in a text and choose the correct relations between the noun phrases that refer to them. Second, s/he has to identify the function and orientation intended by the sender. In the NaviLire project, navigation operations assisting the student are defined used Sextant and the texts are manually annotated by a text linguistics expert.

5.2 Navigation as an alternative to automatic summarization

Many automatic summarization systems have been proposed (Mani, 2001; Minel, 2003). All these systems, based on the principle of phrase, proposition or group extraction, have been confronted to two problems intrinsic to the extraction procedure: i) the rupture of text cohesion, like in cases of anaphora where the corresponding discourse referent is missing; ii) the adaptation of the summary to reader specific needs. Actually, there are no completely satisfying solutions to these problems. An alternative approach is to consider the summarizing process as a reading course belonging to the reader (Crispino and Couto, 2004). Thereby, instead of extracting text fragments, we propose specific reading courses, whose specifications are based on propositions of (Endres-Niggermeyer *et al.*, 1995) and on practice observations made in the frame of the automatic summarization system SERAPHIN evaluation (Minel *et al.*, 1997) and the Filtext framework (Minel *et al.*, 2001).

These works showed that professional summarizers are interested by discourse categories that they retrieve by exploiting text organization and lexical markers. They also showed that these professionals navigate through texts using heuristics acquired by experience. For example, they begin by reading the conclusion, and then they continue by looking, in the introduction, for nominal groups that occurred in the conclusion. This is the knowledge we have modeled with Sextant.

A specific reading course specifies, on the one hand, the kind of discursive category searched by a reader (*e.g.* a conclusion, a definition, an argument, a hypothesis, etc.[1]) and on the other hand, the course in which the segments that linguistically enunciate these categories (typically phrases) must be presented to the reader.

To carry out these reading courses, it is necessary to locate the discursive categories involved and mark them in the text. For this purpose, we used ContextO (Minel *et al.*, 2001). A reading course example is presented in Fig. 3. The reading point is positioned over the first TU, a phrase in this case, annotated "Thematic Announcement". When the user clicks over the TU, NaviTexte recommends her/him four navigation operations. The first one suggests bringing her/him to the following "Thematic Announcement". The others ones suggest going to the first "Conclusion", the first "Recapitulation" and the first "Argument". For a given TU, each navigation operation available has three

[1]For more details on different categories or on what empirical basis were these types derived, see (Minel et al., 2001).

possible states (and there is a visual semantic associated to states), depending if it has been executed (bold font and "*" prefix) or not (normal font and no prefix), and if it exists a TU target (clickable menu option) or not (non-clickable menu option).

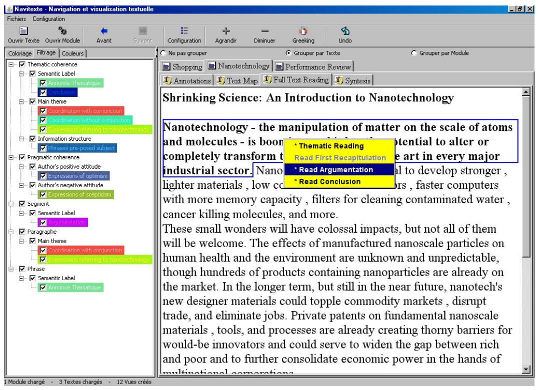


Figure 3. Automatic summarization courses.

These kinds of suggestions (*i.e.* showing available navigation operations for a TU) are made all over the reading process. Consequently, along her/his reading continuum, the reader is assisted by the display of a specific set of signs, and there is no rupture of cohesion because s/he is able to continually see all the text (Battistelli and Minel, 2006).

6 Evaluations

There are few studies of adaptive navigation in hypermedia systems and most of them are focused in measures such as the number of visited nodes or the task completion time (Höök and Svensson, 1999). Are we interested in this kind of measures? Being NaviTexte a generic text navigation system, we think that what it has to be evaluated are the different applications. Each application requires pertinent measures. For example, in NaviLire, the number of nodes or the time factor seems less useful that the comprehension of the text analyzed.

So far, NaviLire has been put into practice on a small scale only, viz. in the teaching of French texts and text linguistics to Danish language students in the 4th year of Language and Communication studies at the Copenhagen Business School. A pilot experiment was carried out in order to evaluate the effects of using the program.

The first results are based on forty answers, of which 35 concern questions about the content of the text. These results show that the *navilistes* (people using NaviLire) have a better comprehension performance than the *papiristes* (people using paper and pencil) for 14 questions, an identical performance for 16 other questions, and a poorer performance for 5 questions (*cf.* Table 1).

	#questions	%
Navilistes better than *Papiristes*	14	40
Navilistes the same as *Papiristes*	16	45,7
Navilistes worse than *Papiristes*	5	14,3
Total	35	100

Table 1. Comparison of *navilistes* and *papiristes* (Lundquist *et al.*, 2006)

Evaluations of the alternative automatic summarization approach are ongoing. Our main problem is that automatic summarization evaluations, well known as difficult to carry out, typically compare to summaries made by professional summarizers (Mani, 2001; Minel, 2003). On the one hand, since we do not create a summary, we do not have an *object* to compare. On the other hand, since we have modeled the professional heuristics, we cannot compare the behavior of our system to theirs because it is exactly what it has been modeled.

7 Conclusions and future work

We have presented our approach to text navigation conceived as a cognitive process that exploits linguistic information present in texts. We have defined it and explained the main differences with the hypertext navigation approach. The four elements needed to implement our approach are described: a text representation, the navigation knowledge modeling language Sextant, the knowledge encoding agents (via applications) and the NaviTexte system.

Two distinct applications of NaviTexte have been presented, showing the versatility of our approach. The quantitative results of our experimentation with Danish students learning French confirm the improvement obtained by using text navigation.

A long term consequence of modeling navigational knowledge is the creation of knowledge bases exchangeable and reusable. Actual collaborations are reusing the knowledge coming from the NaviLire project into others e-learning projects.

We think that our approach may have a significant impact on the way text is being read when its amount or nature does not allow sequential reading (*e.g.* the Web). Related to last works in Web Wise, we plan to couple our approach to Semantic Web approaches to exploit existing annotations.

Acknowledgments

NaviTexte is supported by a grant (U05H01) from Ecos-Sud Program.

References

Battistelli D. and Minel J.-L. 2006. Les systèmes de résumé automatique: comment assurer une continuité référentielle dans la lecture des textes, *in Sabah (Ed.), Compréhension des langues et interaction*, 295-330.

Battistelli D., Minel J-L and Schwer S. 2006, Représentation des expressions calendaires dans les textes: vers une application à la lecture assistée de biographies. *Revue TAL,* 47(3): 1-26.

Benyon D. and Murray D. 1993. Developing Adaptive Systems to Fit Individual Aptitudes, In W.D.Gray, W.E., Helfley and D.Murray (eds.), Proceedings of the 1993 International Workshop on IUI, 115-122. Orlando, FL., New York, ACM Press.

Bilhaut F., Ho-Dac M., Borillo A., Charnois T., Enjalbert P., Le Draoulec A., Mathet Y., Miguet H., Pery-Woodley M.P. and Sarda L. 2003. Indexation discursive pour la navigation. intradocumentaire: cadres temporels et spatiaux dans l'information géographique. Actes de TALN 2003, 315-320.

Bodner R. and Chignell M. 1999. Dynamic hypertext: querying and linking, *ACM Computing Surveys,* 31(4): 120-132.

Boyle C. and Encarnacion A. 1994. MetaDoc: An Adaptive Hypertext Reading System, *UMUAI,* 4: 1-19.

Brusilovsky P. 1996. Methods and techniques of adaptive hypermedia . *UMUAI,* 6(2-3): 87-129.

Brusilovsky P. and Pesin L. 1994. ISIS-Tutor: An adaptive hypertext learning environment. In H.Ueono & V.Stefanuk (eds.), Proceedings of JCKBSE'94.

Brusilovsky P., Schwartz E. and Weber G. 1996. ELM-ART: An Intelligent Tutoring System on World Wide Web, ITS'96, Berlin, Springer, 261-269.

Couto J. 2006. Modélisation des connaissances pour une navigation textuelle assistée. La plate-forme logicielle NaviTexte. PhD, Université Paris-Sorbonne.

Couto J., Lundquist L., Minel J.-L. 2005. Naviguer pour apprendre. *EIAH 2005*, Montpellier, 45-56.

Couto J. and Minel J-L. 2006. Navigation textuelle: Représentation des textes et des connaissances, *Revue TAL*, 47(2): 1-24, Hermès, Paris.

Crispino G. and Couto J. 2004. Construction automatique de résumés. Une approche dynamique. *Revue TAL*, 45(1): 95-120, Hermès, Paris.

Cunningham, H., Maynard, D., Bontcheva, K. and Tablan, V. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications, ACL'02, ACM Press, 168-175.

Danielson D.R. 2002. Web navigation and the behavorial effects of constantly visible maps, Interacting with Computers, 14: 601-618.

Edwards D.M. and Hardman L. 1989. Lost in hyperspace: cognitive mapping and navigation in a hypertext environment, in Hypertext: Theory and Practice. Oxford, Intellect Books, 105-125.

Endres-Niggemeyer B., Maier E. and Sigel A. 1995. How to implement a naturalistic model of abstracting: four core working steps of an expert abstractor, Information Processing et Management, 31(5): 631-674

Géry M. 2002. Un modèle d'hyperdocument en contexte pour la recherche d'information structurée sur le Web. *Revue des Sciences et Technologies de l'Information,* 7/2002*,* Hermès, Paris, 11-44.

Höök K. and Svensson M. 1999. Evaluating Adaptive Navigation Support. In: Social Navigation of Information Space. Springer Verlag, 240-251.

Kaplan C., Fenwick J. and Chen J. 1993. Adaptive Hypertext Navigation Based On User Goals and Context, *UMUAI*, 3, 193-220.

Lundquist L. 1980. *La cohérence textuelle, syntaxe, sémantique, pragmatique*, Copenhagen, Nordisk Forlag.

Lundquist L. 2005. Noms, verbes et anaphores (in)fidèles. Pourquoi les Danois sont plus fidèles que les Français. Langue française. Vol 145: 73-92.

Lundquist L., Minel J.L. and Couto J. 2006. NaviLire, Teaching French by Navigating in Texts, IPMU'2006, Paris.

Mani I. 2001. *Automatic Summarization*, Amsterdam, John Benjamins Publishing Company.

Mathe N. and Chen J. 1994. A User-Centered Approach to Adaptive Hypertext based on an Information Relevance Model, UM'94, Hyannis, MA, 107-114.

Minel J.-L, Cartier E., Crispino G., Desclés J.P., Ben Hazez S. and Jackiewicz A. 2001. Résumé automatique par filtrage sémantique d'informations dans des textes, Présentation de la plate-forme FilText. Technique et Science Informatiques, n°3: 369-396.

Minel J-L. 2003. *Filtrage sémantique. Du résumé à la fouille de textes*. Paris, Hermès.

Minel, J-L., Nugier, S. and Piat, G. 1997. How to appreciate the Quality of Automatic Text Summarization. EACL 97, Madrid, 25-30.

Synset Assignment for Bi-lingual Dictionary with Limited Resource

Virach Sornlertlamvanich
Thatsanee Charoenporn
Chumpol Mokarat
Thai Computational Linguistics Lab.
NICT Asia Research Center,
Thailand Science Park,
Pathumthani, Thailand
{virach,thatsanee,chumpol}@tcllab.org

Hitoshi Isahara
National Institute of Information
and Communications Technology
3-5 Hikaridai, Seika-cho, soraku-gaun,
Kyoto, Japan 619-0289
isahara@nict.go.jp

Hammam Riza
IPTEKNET, Agency for the Assessment and Application of Technology,
Jakarta Pusat 10340, Indonesia
hammam@iptek.net.id

Purev Jaimai
Center for Research on Language
Processing, National University of
Mongolia, Ulaanbaatar, Mongolia
purev@num.edu.mn

Abstract

This paper explores an automatic WordNet synset assignment to the bi-lingual dictionaries of languages having limited lexicon information. Generally, a term in a bilingual dictionary is provided with very limited information such as part-of-speech, a set of synonyms, and a set of English equivalents. This type of dictionary is comparatively reliable and can be found in an electronic form from various publishers. In this paper, we propose an algorithm for applying a set of criteria to assign a synset with an appropriate degree of confidence to the existing bi-lingual dictionary. We show the efficiency in nominating the synset candidate by using the most common lexical information. The algorithm is evaluated against the implementation of Thai-English, Indonesian-English, and Mongolian-English bi-lingual dictionaries. The experiment also shows the effectiveness of using the same type of dictionary from different sources.

1 Introduction

The Princeton WordNet (PWN) (Fellbaum, 1998) is one of the most semantically rich English lexical databases that are widely used as a lexical knowledge resource in many research and development topics. The database is divided by part of speech into noun, verb, adjective and adverb, organized in sets of synonyms, called synset, each of which represents "meaning" of the word entry.

Though WordNet was already used as a starting resource for developing many language WordNets, the construction of the WordNet for any languages can be varied according to the availability of the language resources. Some were developed from scratch, and some were developed from the combination of various existing lexical resources. Spanish and Catalan WordNets, for instance, are automatically constructed using hyponym relation, monolingual dictionary, bilingual dictionary and taxonomy (Atserias et al., 1997). Italian WordNet (Magnini et al., 1994) is semi-automatically constructed from definition in monolingual dictionary, bilingual dictionary, and WordNet glosses. Hungarian WordNet uses bilingual dictionary, monolingual explanatory dictionary, and Hungarian thesaurus in the construction (Proszeky et al., 2002), etc.

This paper presents a new method particularly to facilitate the WordNet construction by using the existing resources having only English equivalents and the lexical synonyms. Our proposed criteria and algorithm for application are evaluated by implementing to Asian languages which occupy quite different language phenomena in terms of grammars and word unit.

To evaluate our criteria and algorithm, we use the PWN version 2.1 containing 207,010 senses classified into adjective, adverb, verb, and noun. The basic building block is a "synset" which is essentially a context-sensitive grouping of synonyms which are linked by various types of relation such as hyponym, hypernymy, meronymy, antonym, attributes, and modification. Our approach is conducted to assign a synset to a lexical entry by considering its English equivalent and lexical synonyms. The degree of reliability of the assignment is defined in terms of confidence score (CS) based on our assumption of the membership of the English equivalent in the synset. A dictionary from different source is also a reliable source to increase the accuracy of the assignment because it can fulfill the thoroughness of the list of English equivalent and the lexical synonyms.

The rest of this paper is organized as follows: Section 2 describes our criteria for synset assignment. Section 3 provides the results of the experiments and error analysis on Thai, Indonesian, and Mongolian. Section 4 evaluates the accuracy of the assignment result, and the effectiveness of the complimentary use of a dictionary from different sources. Section 5 shows a collaborative interface for revising the result of synset assignment. And Section 6 concludes our work.

2 Synset Assignment

A set of synonyms determines the meaning of a concept. Under the situation of limited resources on a language, English equivalent word in a bi-lingual dictionary is a crucial key to find an appropriate synset for the entry word in question. The synset assignment criteria described in this Section relies on the information of English equivalent and synonym of a lexical entry, which is most commonly encoded in a bi-lingual dictionary.

Synset Assignment Criteria

Applying the nature of WordNet which introduces a set of synonyms to define the concept, we set up four criteria for assigning a synset to a lexical entry. The confidence score (CS) is introduced to annotate the likelihood of the assignment. The highest score, CS=4, is assigned to the synset that is evident to include more than one English equivalent of the lexical entry in question. On the contrary, the lowest score, CS=1, is assigned to any synset that occupies only one of the English equivalents of the lexical entry in question when multiple English equivalents exist.

The details of assignment criteria are elaborated as in the followings. L_i denotes the lexical entry, E_j denotes the English equivalent, S_k denotes the synset, and $\in$ denotes the member of a set:

Case 1: Accept the synset that includes more than one English equivalent with confidence score of 4.

Figure 1 simulates that a lexical entry L_0 has two English equivalents of E_0 and E_1. Both E_0 and E_1 are included in a synset of S_1. The criterion implies that both E_0 and E_1 are the synset for L_0 which can be defined by a greater set of synonyms in S_1. Therefore the relatively high confidence score, CS=4, is assigned for this synset to the lexical entry.

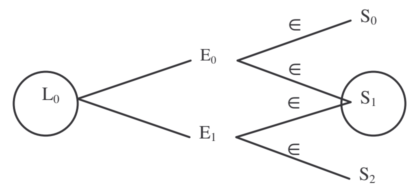


Figure 1. Synset assignment with SC=4

Example:

L_0: เป้าหมาย

E_0: aim E_1: target

S_0: purpose, intent, intention, **aim**, design

S_1: **aim**, object, objective, **target**

S_2: **aim**

In the above example, the synset, S_1, is assigned to the lexical entry, L_0, with CS=4.

Case 2: Accept the synset that includes more than one English equivalent of the synonym of the lexical entry in question with confidence score of 3.

In case that Case 1 fails in finding a synset that includes more than one English equivalent, the English equivalent of a synonym of the lexical entry is picked up to investigate.

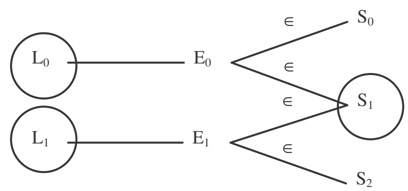


Figure 2. Synset assignment with SC=3

Figure 2 simulates that an English equivalent of a lexical entry L_0 and its synonym L_1 are included in a synset S_1. In this case the synset S_1 is assigned to both L_0 and L_1 with CS=3. The score in this case is lower than the one assigned in Case 1 because the synonym of the English equivalent of the lexical entry is indirectly implied from the English equivalent of the synonym of the lexical entry. The newly retrieved English equivalent may not be distorted.

Example:

L_0: จ้อง L_1: เพ่งมอง

E_0: stare E_1: gaze

S_0: **gaze, stare** S_1: **stare**

In the above example, the synset, S_0, is assigned to the lexical entry, L_0, with CS=3.

Case 3: Accept the only synset that includes the only one English equivalent with confidence score of 2.

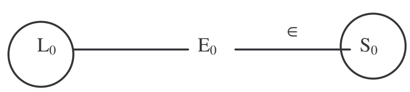


Figure 3. Synset assignment with SC=2

Figure 3 simulates the assignment of CS-2 when there is only one English equivalent and there is no synonym of the lexical entry. Though there is no any English equivalent to increase the reliability of the assignment, in the same time there is no synonym of the lexical entry to distort the relation. In this case, the only one English equivalent shows it uniqueness in the translation that can maintain a degree of the confidence.

Example:

L_0: สูติแพทย์ E_0: obstetrician

S_0: **obstetrician**, accoucheur

In the above example, the synset, S_0, is assigned to the lexical entry, L_0, with CS=2.

Case 4: Accept more than one synset that includes each of the English Equivalent with confidence score of 1.

Case 4 is the most relax rule to provide some relation information between the lexical entry and a synset. Figure 4 simulates the assignment of CS=1 to any relations that do not meet the previous criteria but the synsets that include one of the English equivalent of the lexical entry.

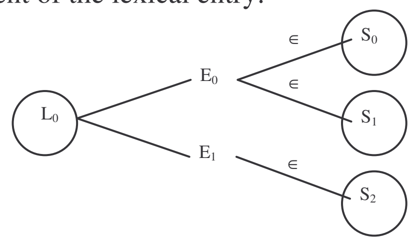


Figure 4. Synset assignment with SC=1

Example:

L_0: ช่อง

E_0: hole E_1: canal

S_0: **hole**, hollow

S_1: **hole**, trap, cakehole, maw, yap, gop

S_2: **canal**, duct, epithelial duct, channel

In the above example, each synset, S_0, S_1, and S_2 is assigned to lexical entry L_0, with CS=1.

3 Experiment results

We applied the synset assignment criteria to a Thai-English dictionary (MMT dictionary) (CICC, 1995) with the synset from WordNet 2.1. To compare the ratio of assignment for Thai-English dictionary, we also investigate the synset assignment of Indonesian-English and Mongolian-English dictionaries.

	WordNet (synset)		T-E Dict (entry)	
	total	assigned	total	assigned
Noun	145,103	18,353 (13%)	43,072	11,867 (28%)
Verb	24,884	1,333 (5%)	17,669	2,298 (13%)
Adjective	31,302	4,034 (13%)	18,448	3,722 (20%)
Adverb	5,721	737 (13%)	3,008	1,519 (51%)
total	207,010	24,457 (12%)	82,197	19,406 (24%)

Table 1. Synset assignment to T-E dictionary

In our experiment, there are only 24,457 synsets from 207,010 synsets, which is 12% of the total number of the synset that can be assigned to Thai lexical entries. Table 1 shows the successful rate in assigning synset to Thai-English dictionary. About 24 % of Thai lexical entries are found with the English equivalents that meet one of our criteria.

Going through the list of unmapped lexical entry, we can classify the errors into three groups:-

1. Compound

   The English equivalent is assigned in a com-

pound, especially in case that there is no an appropriate translation to represent exactly the same sense. For example,

L: ร้านค้าปลีก E: retail shop

L: กระชาก E: pull sharply

2. Phrase

   Some particular words culturally used in one language may not be simply translated into one single word sense in English. In this case, we found it explained in a phrase. For example,

   L: ร้านสวด

   E: small pavilion for monks to sit on to chant

   L: กรรเจียก

   E: bouquet worn over the ear

3. Word form

   Inflected forms i.e. plural, past participle, are used to express an appropriate sense of a lexical entry. This can be found in non-inflection languages such as Thai and most of Asian languages. For example,

   L: ร้าวระทมใจ E: grieved

The above English expressions cause an error in find an appropriate synset.

	WordNet (synset)		I-E Dict (entry)	
	total	assigned	total	assigned
Noun	145,103	4,955 (3%)	20,839	2,710 (13%)
Verb	24,884	7,841 (32%)	15,214	4,243 (28%)
Adjective	31,302	3,722 (12%)	4,837	2,463 (51%)
Adverb	5,721	381 (7%)	414	285 (69%)
total	207,010	16,899 (8%)	41,304	9,701 (24%)

Table 2. Synset assignment to I-E dictionary

We applied the same algorithm to Indonesia-English and Mongolian-English (Hangin, 1986) dictionaries to investigate how it works with other languages in terms of the selection of English equivalents. The difference in unit of concept is basically understood to effect the assignment of English equivalents in bi-lingual dictionaries. In Table 2, the size of Indonesian-English dictionary is about half of Thai-English dictionary. The success rates of assignment to the lexical entry are the same but the rate of synset assignment of Indonesian-English dictionary is lower than one of Thai-English dictionary. This is because the total number of lexical entry is almost in the half size.

	WordNet (synset)		ME Dict (entry)	
	total	assigned	Total	assigned
Noun	145,103	268 (0.18%)	168	125 (74.40%)
Verb	24,884	240 (0.96%)	193	139 (72.02%)
Adjective	31,302	211 (0.67%)	232	129 (55.60%)
Adverb	5,721	35 (0.61%)	42	17 (40.48%)
total	207,010	754 (0.36%)	635	410 (64.57%)

Table 3. Synset assignment to M-E dictionary

A small set of Mongolian-English dictionary is also evaluated. Table 3 shows the result of synset assignment.

These experiments show the effectiveness of using English equivalents and synonyms information from limited resources in assigning WordNet synsets.

4 Evaluations

In the evaluation of our approach for synset assignment, we randomly selected 1,044 synsets from the result of synset assignment to Thai-English dictionary (MMT dictionary) for manually checking. The random set covers all types of part-of-speech and degrees of confidence score (CS) to confirm the approach in all possible situations. According to the supposition of our algorithm that the set of English equivalents of a word entry and its synonyms are significant information to relate to a synset of WordNet, the result of accuracy will be correspondent to the degree of CS. The detail number of synsets to be used in the evaluation is shown in Table 4.

	CS=4	CS=3	CS=2	CS=1	total
Noun	7	479	64	272	822
Verb		44	75	29	148
Adjective	1	25		32	58
Adverb	7	4	4	1	16
total	15	552	143	334	1044

Table 4. Random set of synset assignment

Table 5 shows the accuracy of synset assignment by part-of-speech and CS. A small set of adverb synsets are 100% correctly assigned irrelevant to its CS. The total number of adverbs for the evaluation could be too small. The algorithm shows a better result of 48.7% in average for noun

synset assignment and 43.2% in average for all part-of-speech.

	CS=4	CS=3	CS=2	CS=1	total
Noun	5 (71.4%)	306 (63.9%)	34 (53.1%)	55 (20.2%)	400 (48.7%)
Verb		23 (52.3%)	6 (8.0%)	4 (13.8%)	33 (22.3%)
Adjective		2 (8.0%)			2 (3.4%)
Adverb	7 (100%)	4 (100%)	4 (100%)	1 (100%)	16 (100%)
total	12 (80.0%)	335 (60.7%)	44 (30.8%)	60 (18%)	451 (43.2%)

Table 5. Accuracy of synset assignment

With the better information of English equivalents marked with CS=4, the assignment accuracy is as high as 80.0% and decreases accordingly due to the CS value. This confirms that the accuracy of synset assignment strongly relies on the number of English equivalents in the synset. The indirect information of English equivalents of the synonym of the word entry is also helpful. It yields 60.7% of accuracy in synset assignment for the group of CS=3. Others are quite low but the English equivalents are somehow useful to provide the candidates for expert revision.

	CS=4	CS=3	CS=2	CS=1	total
Noun	2		22	29	53
Verb		2	6	4	12
Adjective					
Adverb					
total	2	2	28	33	65

Table 6. Additional correct synset assignment by other dictionary (LEXiTRON)

To examine the effectiveness of English equivalent and synonym information from different source, we consulted another Thai-English dictionary (LEXiTRON). Table 6 shows the improvement of the assignment by the increased number of correct assignment in each type. We can correct more in noun and verb but not adjective. Verb and adjective are ambiguously defined in Thai lexicon, and the number of the remained adjective is too few, therefore, the result should be improved unconcerned with the type.

	CS=4	CS=3	CS=2	CS=1	total
total	14 (93.3%)	337 (61.1%)	72 (50.3%)	93 (27.8%)	516 (49.4%)

Table 7. Improved correct synset assignment by additional bi-lingual dictionary (LEXiTRON)

Table 7 shows the total improvement of the assignment accuracy when we integrated English equivalent and synonym information from different source. The accuracy for synsets marked with CS=4 is improved from 80.0% to 93.3% and the average accuracy is also significantly improved from 43.2% to 49.4%. All types of synset are significantly improved only if a bi-lingual dictionary from different sources is available.

5 Collaborative Work on Asian WordNet

There are some efforts in developing WordNets of some Asian languages, e.g. Chinese, Japanese, Korean (Choi, 2003), (Choi et al., 2004), (Kaji et al., 2006), (KorLex, 2006), (Huang, 2007) and Hindi (Hindi Wordnet, 2007). The number of languages that have been successfully developed their WordNets is still limited to some active research in this area. However, the extensive development of WordNet in other languages is important, not only to help in implementing NLP applications in each language, but also in inter-linking WordNets of different languages to develop multi-lingual applications to overcome the language barrier.

We adopt the proposed criteria for automatic synset assignment for Asian languages which has limited language resources. Based on the result from the above synset assignment algorithm, we provide KUI (Knowledge Unifying Initiator) (Sornlertlamvanich, 2006), (Sornlertlamvanich et al., 2007) to establish an online collaborative work in refining the WorNets.

KUI is a community software which allows registered members including language experts revise and vote for the synset assignment. The system manages the synset assignment according to the preferred score obtained from the revision process. As a result, the community WordNets will be accomplished and exported into the original form of WordNet database. Via the synset ID assigned in the WordNet, the system can generate a cross language WordNet result. Through this effort, an initial version of Asian WordNet can be fulfilled.

Figure 5 illustrates the translation page of KUI[1]. In the working area, the login member can participate in proposing a new translation or vote for the preferred translation to revise the synset assignment. Statistics of the progress as well as many useful functions such as item search, record jump, chat, list of online participants are also provided.

[1] http://www.tcllab.org/kui

KUI is actively facilitating members in revising the Asian WordNet database.

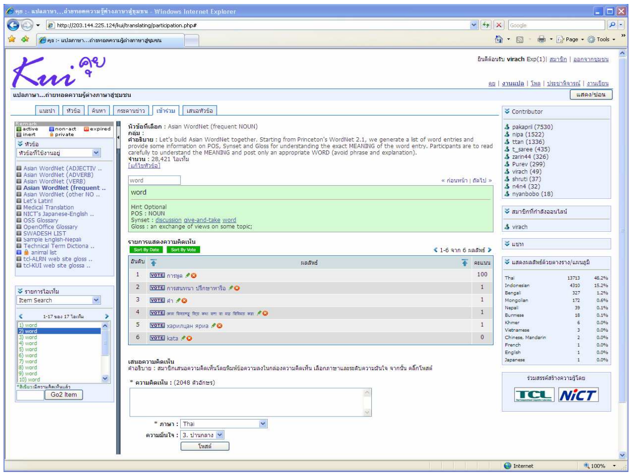


Figure 5. Sample of KUI interface

6 Conclusion

Our synset assignment criteria were effectively applied to languages having only English equivalents and its lexical synonym. Confidence score was proved efficiently assigned to determine the degree of reliability of the assignment which later was a key value in the revision process. Languages in Asia are significantly different from the English language in terms of grammar and lexical word unit. The differences prevent us from finding the target synset by following just the English equivalent. Synonyms of the lexical entry and additional dictionary from different sources can be complimentarily used to improve the accuracy in the assignment. Applying the same criteria to other Asian languages also yielded a satisfactory result. Following the same process that we had implemented to the Thai language, we are expecting an acceptable result from the Indonesian, Mongolian languages and so on. After the revision at KUI, the initial stage of Asian WordNet will be referable through the assigned synset ID.

References

Bernardo Magnini, Carlo Strapparava, Fabio Ciravegna and Emanuele Pianta, 1994. *A Project for the Construction of an Italian Lexical Knowledge Base in the Framework of WordNet*, IRST Technical Report # 9406-15.

Chu-Ren Huang, 2007. *Chinese Wordnet*, Academica Sinica, Available at http://bow.sinica.edu.tw/wn/

CICC. 1995. *Thai Basic Dictionary: Technical Report*, Japan.

Fellbaum, Christiane (ed.), 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Gabor Proszeky, Marton Mihaltz, 2002. *Semi-Automatic Development of the Hungarian WordNet*, Proceedings of the LREC 2002, Spain

Gombojab Hangin with John R.Krueger and Paul D.Buell, William V.Rozycki, Robert G.Service, 1986. *A modern Mongolian-English dictionary*. Indiana University, Research Institute for Inner Asian Studies.

Hindi Wordnet, 2007. Available at http://www.cfilt.iitb.ac.in/wordnet/webhwn/

Hiroyuki Kaji and Mariko Watanabe, 2006. *Automatic Construction of Japanese WordNet*, Proceedings of LREC2006, Italy.

J. Atserias, S. Clement, X. Farreres, German Rigau, H. Rodríguez, 1997. *Combining Multiple Methods for the Automatic Construction of Multilingual WordNets*, Proceedings of the International Conference on Recent Advances in Natural Language, Bulgaria.

K.S. Choi, H.S. Bae, W.Kang, J. Lee, E. Kim, H. Kim, D. Kim, Y. Song1, and H. Shin, 2004. *Korean-Chinese-Japanese Multilingual Wordnet with Shared Semantic Hierarchy*, Proceediongs of LREC 2004, Portugal.

Key-Sun Choi, 2003. *CoreNet: Chinese-Japanese-Korean wordnet with shared semantic hierarchy*, Proceedings of Natural Language Processing and Knowledge Engineering, Beijing.

Korlex, 2006. *Korean WordNet*, Korean Language Processing Lab, Pusan National University, 2007. Available at http://164.125.65.68/

NECTEC, 2006. *LEXiTRON: Thai-English Dictionary*, Available at http://lexitron.nectec.or.th/

Spanish and Catalan WordNets, 2006. Available at http://www.lsi.upc.edu/~nlp/

Virach Sornlertlamvanich, 2006. *KUI: The OSS-Styled Knowledge Development System*, Proceedings of The 7th AOSS Symposium, Malaysia.

Virach Sornlertlamvanich, Thatsanee Charoenporn, Kergit Robkop, and Hitoshi Isahara. *Collaborative Platform for Multilingual Resource Development and Intercultural Communication*, Proceedings of the First International Workshop on Intercultural Collaboration (IWIC2007), Japan.

Ranking Words for Building a Japanese Defining Vocabulary

Tomoya Noro
Department of Computer Science
Tokyo Institute of Technology
2-12-1 Meguro, Tokyo, 152-8552 Japan
noro@tt.cs.titech.ac.jp

Takehiro Tokuda
Department of Computer Science
Tokyo Institute of Technology
2-12-1 Meguro, Tokyo, 152-8552 Japan
tokuda@cs.titech.ac.jp

Abstract

Defining all words in a Japanese dictionary by using a limited number of words (defining vocabulary) is helpful for Japanese children and second-language learners of Japanese. Although some English dictionaries have their own defining vocabulary, no Japanese dictionary has such vocabulary as of yet. As the first step toward building a Japanese defining vocabulary, we ranked Japanese words based on a graph-based method. In this paper, we introduce the method, and show some evaluation results of applying the method to an existing Japanese dictionary.

1 Introduction

Defining all words in a dictionary by using a limited number of words (defining vocabulary) is helpful in language learning. For example, it would make it easy for children and second-language learners to understand definitions of all words in the dictionary if they understand all words in the defining vocabulary. In some English dictionaries such as the Longman Dictionary of Contemporary English (LDOCE) (Proctor, 2005) and the Oxford Advanced Learner's Dictionary (OALD) (Hornby and Ashby, 2005), 2,000-3,000 words are chosen and all headwords are defined by using the vocabulary. Such dictionaries are widely used for language learning.

Currently, however, such a dictionary in which a defining vocabulary is specified has not been available in Japanese. Although many studies for Japanese "basic vocabulary" have been done (National Institute for Japanese Language, 2000), "basic vocabulary" in the studies means a vocabulary which children or second-language learners have (or should learn). In other words, the aim of such studies is to determine a set of headwords which should be included in a Japanese dictionary for children or second-language learners.

We think that there is a difference between "defining vocabulary" and "basic vocabulary". Although basic vocabulary is usually intended for learning expression in newspaper/magazine articles, daily conversation, school textbook, etc, a defining vocabulary is intended for describing word definition in a dictionary. Some words (or phrases) which are often used in word definition, such as "... の略 (abbreviation of ...)", "転じて (change/shift)" [1], "物事 (thing/matter)" etc, are not included in some kinds of basic vocabulary. Additionally only one word in a set of synonyms should be included in a defining vocabulary even if all of them are well-known. For example, if a word "使う (use)" is included in a defining vocabulary, synonyms of the word, such as "使用する", "利用する" and "用いる", are not needed.

A goal of this study is to try to build a Japanese defining vocabulary on the basis of distribution of words used in word definition in an existing Japanese dictionary. In this paper, as the first step of this, we introduce the method for ranking Japanese words, and show some evaluation results of applying the method to an existing Japanese dictionary. Also, we compare the results with two kinds of basic vo-

[1] It is a kind of conjunction used to describe a new meaning comes out of the original meaning.

Headword	Word definition
硬貨 (kouka)	金属製の貨幣。
紙幣 (shihei)	金属貨幣の代用として流通する紙の貨幣。
外貨 (gaika)	外国の貨幣。
贋金 (nisegane)	にせの貨幣（特に、硬貨）。

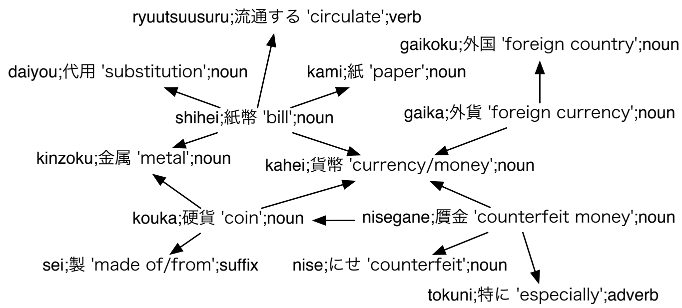


Figure 1: A word reference graph

cabulary, and discuss the difference.

2 Related Work

Kasahara et al. constructed a Japanese semantic lexicon, called “Lexeed” (Kasahara et al., 2004). The lexicon contains the most familiar 28,000 Japanese words, which are determined through questionnaires. All words in the lexicon are defined by using 16,900 words in the same lexicon. However, the size of the vocabulary seems to be too large compared to the size of the defining vocabularies used in LDOCE and OALD. We also think that whether a word is familiar or not does not always correspond to whether the word is necessary for word definition or not.

Gelbukh et al. proposed a method for detecting cycles in word definitions and selecting primitive words (Gelbukh and Sidorov, 2002). This method is intended for converting an existing “human-oriented” dictionary into a “computer-oriented” dictionary, and the primitive words are supposed not to be defined in the dictionary.

Fukuda et al. adopted an LSA-based (latent semantic analysis) method to build a defining vocabulary (Fukuda et al., 2006). The method would be another solution to this issue although only a small evaluation experiment was carried out.

3 Method

Our method for building a Japanese defining vocabulary is as follows:

1. For each headword in an existing Japanese dictionary, represent the relationship between the headword and each word in the word definition as a directed graph (word reference graph).
2. Compute the score for each word based on the word reference graph.
3. Nominate the high ranked words for the Japanese defining vocabulary.
4. Manually check whether each nominated word is appropriate as defining vocabulary or not, and remove the word if it is not appropriate.

In the rest of this section, we introduce our method for constructing word reference graph and computing score for each word.

3.1 Word Reference Graph

A word reference graph is a directed graph representing relation between words. For each headword in a dictionary, it is connected to each word in the word definition by a directed edge (Figure 1). Nodes in the graph are identified by reading, base form (orthography), and parts-of-speech because some words have more than one part-of-speech or reading (“余り (the reading is ‘*amari*’)” has two parts-of-speech, noun and adverb, and “小節” has two readings, “*shousetsu*” and “*kobushi*”). Postpositions, auxiliary verbs, numbers, proper names, and symbols are removed from the graph.

3.2 Computing The Score for Each Word

The score of each word is computed under the assumption that

1. A score of a word which appears in many word definitions will be high.
2. A score of a word which appears in the definition of a word with high score will also be high.

If a word is included in a defining vocabulary, words in the word definition may need to be included in order to define the word. The second assumption reflects the intuition. We adopt the algorithm of PageRank (Page et al., 1998) or LexRank (Erkan and Radev, 2004), which computes the left eigenvector of the adjacency matrix of the word reference graph with the corresponding eigenvalue of 1.

4 Evaluation

4.1 Experimental Setup

We used the Iwanami Japanese dictionary corpus (Hasida, 2006). The corpus was built by annotating the Iwanami Japanese dictionary (the 5th edition) with the GDA tags (Hasida, 2004) and some other tags specific to the corpus. Although it has many kinds of tags, we focus on information about the headword (hd), orthography (orth), part-of-speech (pos), sentence unit in word definition (su), and morpheme (n, v, ad, etc.). We ignore kind of additional information, such as examples (eg), grammatical explanations (gram), antonyms (ant), etymology (etym), references to other entries (sr), etc, since such information is not exactly "word definition". Words in parentheses, "「」" and "『』", are also ignored since they are used to quote some words or expressions for explanation and should be excluded from consideration of defining vocabulary.

Some problems arose when constructing a word reference graph.

1. **Multiple ways of writing in *kanji*:**

   For example, in the Iwanami Japanese dictionary, "引く", "弾く", "曳く", "牽く", "碾く", "轢く" and "退く" appear in an entry of a verb "*hiku*" as its orthography. If more than one writing way appear in one entry, they are merged into one node in the word reference graph (they are separated if they have different part-of-speech).

2. **Part-of-speech conversion:**

   While each word in word definition was annotated with part-of-speech by corpus annotators, part-of-speech of each headword in the dictionary was determined by dictionary editors. The two part-of-speech systems are different from each other. In order to resolve the difference, we prepared a coarse-grained part-of-speech system (just classifying into noun, verb, adjectives, etc.), and converted part-of-speech of each word.

3. **Word segmentation:**

   In Japanese, words are not segmented by spaces and the word segmentation policy for corpus annotation sometimes disagree with the policy for headword registration of the Japanese Iwanami dictionary. In the case that two consecutive nouns or verbs are in word definition and a word consisting of the two words is included as a headword in the dictionary, the two words are merged into one word.

4. **Difference in writing way between a headword and a word in word definition:**

   In Japanese language, we have three kind of characters, *kanji*, *hiragana*, and *katakana*. Most of the headwords appearing in a dictionary (except loanwords) are written in *kanji* as orthography. On the other hand, for example, "事 (matter)" is usually written in *hiragana* ("こと") in word definition. However, it is difficult to know automatically that a word "こと" in word definition means "事", since the dictionary has other entries which has the same reading "*koto*", such as "琴 (Japanese harp)" and "古都 (ancient city)". We merged two nodes in the word reference graph manually if the two words are the same and only different in the writing way.

As a result, we constructed a word reference graph consisting of 69,013 nodes.

We adopted the same method as (Erkan and Radev, 2004) for computing the eigenvector of the adjacency matrix (score of each word). Damping factor for random walk and error tolerance are set to 0.15 and 10^{-4} respectively.

4.2 Result

Table 1 shows the top-50 words ranked by our method. Scores are normalized so that the score of the top word is 1.

Table 1: The top-50 words

	Score	Reading	Orthography	POS	Meaning
1	1.000	*aru*	有る, 在る	V	exist
2	.7023	*i*	意	N	meaning
3	.6274	*aru*	或る	Adn *	certain/some
4	.5927	*koto*	事	N	matter
5	.5315	*suru*	為る	V	do
6	.3305	*mono*	物, 者	N	thing/person
7	.2400	*sono*	其の	Adn *	its
8	.2118	*hou*	方	N	direction
9	.1754	*tatsu*	立つ, 建つ	V	stand/build
10	.1719	*mata*	又, 復, 亦	Conj	and/or
11	.1713	*iru*	居る, 処る	V	exist
12	.1668	*hito*	人	N	person
13	.1664	*tsukau*	使う, 遣う	V	use
14	.1337	*iku*	行く, 往く	V	go/die
15	.1333	*naru*	成る, 為る 生る	V	become
16	.1324	*iu*	言う, 云う 謂う	V	say
17	.1244	*monogoto*	物事	N	thing/matter
18	.1191	*dou*	同	Adn *	same
19	.1116	*sore*	其れ	Pron	it
20	.1079	*toki*	時, 刻	N	time
21	.1074	*teki*	的	Suffix	-like
22	.1020	*souiu*	そういう	Adn *	such
23	.09682	*joutai*	状態	N	situation
24	.09165	*arawasu*	表す, 現す 顕す, 著す	V	represent/ appear/ write a book
25	.08968	*ieru*	言える	V	can say
26	.08780	*ei*	A	N	A
27	.08585	*ten*	点	N	point
28	.08526	*tokuni*	特に	Adv	especially
29	.08491	*go*	語	N	word
30	.08449	*iiarawasu*	言い表す	V	express
31	.08255	*matawa*	又は	Conj	or
32	.07285	*erabitoru*	選び取る	V	choose & take
33	.07053	*baai*	場合	N	case
34	.06975	*tokoro*	所, 処	N	place
35	.06920	*katachi*	形	N	shape
36	.06873	*nai*	無い	Adj	no
37	.06855	*kotogara*	事柄	N	matter
38	.06709	*bii*	B	N	B
39	.06507	*yakunitatsu*	役に立つ	V	useful
40	.06227	*wareware*	我我	Pron	we
41	.06109	*joshi*	助詞	N	postposition
42	.06089	*iitsukeru*	言いつける	V	tell
43	.06079	*ten*	転	N	change/shift
44	.05989	*eigo*	英語	N	English language
45	.05972	*jibun*	自分	N	self
46	.05888	*kata*	方	Suffix	way
47	.05879	*tame*	為	N	reason/aim
48	.05858	*kaku*	書く, 描く	V	write/draw/ paint
49	.05794	*kangaeru*	考える 勘える	V	think
50	.05530	*fukushi*	副詞	N	adverb

* "Adn" indicates "adnominal word", which is a Japanese-specific category and always modifies nouns.

From the result, we can find that not only common words which may be included in a "basic vocabulary", such as "有る (exist)", "或る (certain/some)" [2], "為る (do')", "物 (thing)", etc., but also words which are not so common but are often used in word definition, such as "意 (meaning)", "物事 (thing/matter)", "転 (change/shift)".

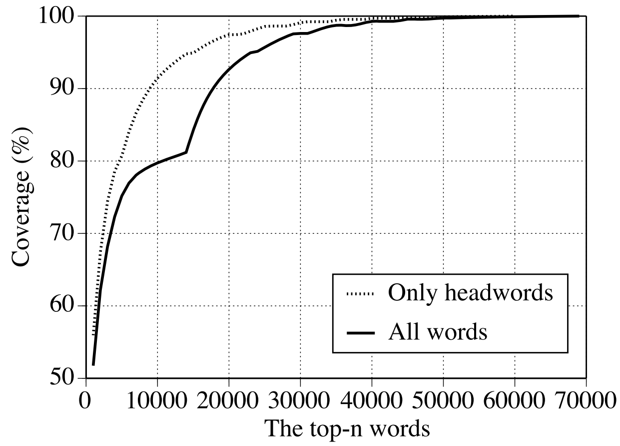


Figure 2: Word coverage

On the other hand, some words in the top ranked words, such as "A" and "B", seem not to be appropriate for defining vocabulary. These words appear only in word definition and are not included in the Iwanami Japanese dictionary as headwords (i.e. unregistered words) [3]. The score of an unregistered word tends to be higher than it should be, since the node corresponding to the word has no edge to other nodes in the word reference graph.

Figure 2 shows word coverage, i.e. percentage of words appearing in word definition which were ranked in the top-n. From the result (solid line), we can find that the increase in coverage around $n = 10,000$ is low and the coverage increases suddenly from $n = 15,000$. This is because all unregistered words were ranked in the top-15000. If all unregistered words are removed, the increase in coverage gets gradually lower as n increases (dotted line).

In construction of a word reference graph, 9,327 words were judged as unregistered words. The reason is as follows:

1. **Part-of-speech mismatch:**

   In order to solve the difference between the part-of-speech system for annotation of headwords and the system for annotation of words in the definition of each headword, we pre-

[2] It is used to say something undetermined or to avoid saying something exactly even if you know that.

[3] In some word definitions, roman letters are used as variables.

pared a coarse-grained part-of-speech system and converted part-of-speech of each word. However, the conversion failed in some cases. For example, some words are annotated with suffix or prefix in word definition, while they are registered as noun in the dictionary.

2. **Mismatch of word segmentation:**

   Two consecutive nouns or verbs in word definition were merged into one word if a word consisting of the two words is included as a headword in the Iwanami Japanese dictionary. However, in the case that a compound word is treated as one word in word definition and the word is not registered as a headword in Iwanami Japanese dictionary, the word is judged as an unregistered word.

3. **Error in format or annotation of the corpus:**

   Since the Iwanami Japanese dictionary corpus has some errors in format or annotation, we removed entries which have such errors before construction of the word reference graph. Headwords which were removed for this reason are judged as unregistered words.

4. **Real unregistered words:**

   Some words in word definition are not registered as headwords actually. For example, although a noun "英語 (English language)" appears in word definition, the word is not registered as a headword.

Unregistered words should carefully be checked whether they are appropriate as defining vocabulary or not at the third step of our method described in section 3.

4.3 Comparison

In order to look at the difference between the result and so-called "basic vocabulary", we compared the result with two types of basic vocabulary: one was built by the National Institute for Japanese Language (including 6,099 words) and the other was built by the Chuo Institute for Educational Research (including 4,332 words) (National Institute for Japanese Language, 2001). These two types of vocabulary are intended for foreigners (second-language learners)

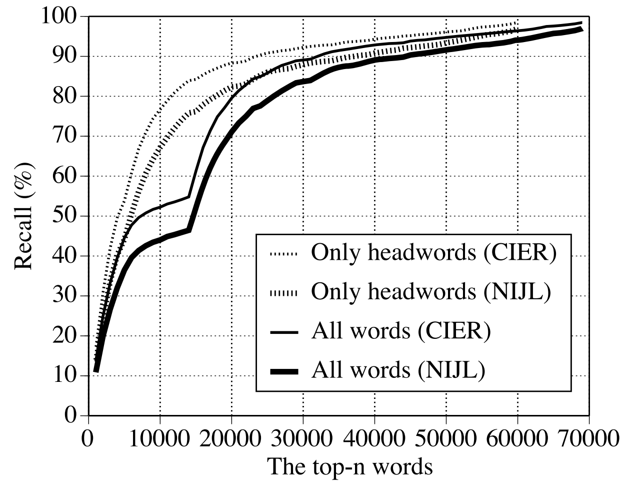


Figure 3: Comparison with two types of basic vocabulary

Table 2: High-ranked words out of the two basic vocabularies

Rank	Reading	Orthography	POS	Meaning
51	*tenjiru*	転じる	V	shift/change
102	*youhou*	用法	N	usage
113	*ryaku*	略	N	abbreviation
372	*furumai*	振舞い	N	behavior
480	*sashishimesu*	指し示す	V	indicate

and Japanese children (elementary school students) respectively.

Figure 3 shows recall, i.e. percentage of the number of words appearing in both our result and each vocabulary out of the number of words in the vocabulary. As in the case of word coverage, the increase in recall around $n = 10,000$ is low if unregistered words are not removed (solid lines). If the same number of headwords as the size of each basic vocabulary are picked up from our result, it can be found that about 50% of the words are shared with each basic vocabulary (dotted lines).

Some of the high-ranked words out of the two basic vocabularies and some of the low-ranked words in the vocabularies are listed in Table 2 and 3. Although it would be natural that the words listed in Table 2 are not included in the basic vocabularies, they are necessary for describing word definition. On the other hand, the words listed in Table 3 may not be necessary for describing word definition, while they are often used in daily life.

Table 3: Low-ranked words in the two basic vocabularies

Rank	Reading	Orthography	POS	Meaning
20062	*taifuu*	台風	N	typhoon
20095	*obaasan*	お婆さん	N	grandmother
31097	*tetsudau*	手伝う	V	help/assist
37796	*kamu*	噛む	V	bite
47579	*mochiron*	勿論	Adv	of course
65413	*tokoroga*	ところが	Conj	but/however

5 Conclusion

In this paper, we introduced the method for ranking Japanese words in order to build a Japanese defining vocabulary. We do not think that a set of the top-n words ranked by our method could be a defining vocabulary as is. The high ranked words need to be checked whether they are appropriate as defining vocabulary or not.

As described in section 1, defining all words with a defining vocabulary is helpful in language learning. In addition, we expect that the style of writing word definitions (e.g. which word should be used, whether the word should be written in *kanji* or *hiragana*, etc.) can be controlled with the vocabulary.

This kind of vocabulary could also be useful for NLP researches as well as language learning. Actually, defining vocabularies used in LDOCE and OALD are often used in some NLP researches.

The future work is the following:

- The size of a defining vocabulary needs to be determined. Although all words in LDOCE or OALD are defined by 2,000-3,000 words, the size of a Japanese defining vocabulary may be larger than English ones.
- Wierzbicka presented the notion of conceptual primitives (Wierzbicka, 1996). We need to look into our result from a linguistic point of view, and to discuss the relation.
- It is necessary to consider how to describe word definition as well as which word should be used for word definition. Definition of each word in a dictionary includes many kinds of information, not only the word sense but also historical background, grammatical issue, etc. Only word sense should be described with a defining vocabulary, since the other information is a little different from word sense and it may be difficult to describe the information with the same vocabulary.

References

Güneş Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Muhtar Fukuda, Yasuhiro Ogawa, and Katsuhiko Toyama. 2006. Automatic generation of dictionary definition words based on latent semantic analysis. In *the 5th Forum on Information Technology*. In Japanese.

Alexander F. Gelbukh and Grigori Sidorov. 2002. Automatic selection of defining vocabulary in an explanatory dictionary. In *the 3rd International Conference on Computational Linguistics and Intelligent Text Processing*, pages 300–303.

Koiti Hasida, 2004. *The GDA Tag Set*. http://i-content.org/GDA/tagset.html.

Koiti Hasida, 2006. *Annotation of the Iwanami Japanese Dictionary – Anaphora, Coreference And Argument Structure –*. http://www.i-content.org/rwcDB /iwanami/doc/tag.html (In Japanese).

A. S. Hornby and Michael Ashby, editors. 2005. *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press.

Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. 2004. Construction of Japanese Semantic Lexicon: Lexeed. In *IPSJ SIGNL 159*, pages 75–82. In Japanese.

The National Institute for Japanese Language, editor. 2000. *Japanese Basic Vocabulary – An Annotated Bibliography And a Study –*. Meiji Shoin. In Japanese.

The National Institute for Japanese Language, editor. 2001. *A Basic Study of Basic Vocabulary for Education – Construction of a Database of Basic Vocabulary for Education –*. Meiji Shoin. In Japanese.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford University.

Paul Proctor, editor. 2005. *Longman Dictionary of Contemporary English*. Longman.

Anna Wierzbicka. 1996. *Semantics: Primes and Universals*. Oxford University Press.

Automatically Identifying Computationally Relevant Typological Features

William D. Lewis*
Microsoft Research
Redmond, WA 98052-6399
wilewis@microsoft.com

Fei Xia
University of Washington
Seattle, WA 98195
fxia@u.washington.edu

Abstract

In this paper we explore the potential for identifying computationally relevant typological features from a multilingual corpus of language data built from readily available language data collected off the Web. Our work builds on previous structural projection work, where we extend the work of projection to building individual CFGs for approximately 100 languages. We then use the CFGs to discover the values of typological parameters such as word order, the presence or absence of definite and indefinite determiners, etc. Our methods have the potential of being extended to many more languages and parameters, and can have significant effects on current research focused on tool and resource development for low-density languages and grammar induction from raw corpora.

1 Introduction

There is much recent interest in NLP in "low-density" languages, languages that typically defy standard NLP methodologies due to the absence or paucity of relevant digital resources, such as treebanks, parallel corpora, machine readable lexicons and grammars. Even when resources such as raw or parallel corpora exist, they often cannot be found of sufficient size to allow the use of standard machine learning methods. In some recent grammar induction and MT work (Haghighi and Klein, 2006; Quirk et al., 2005) it has been shown that even a small amount of knowledge about a language, in the form of grammar fragments, treelets or prototypes, can go a long way in helping with the induction of a grammar from raw text or with alignment of parallel corpora.

In this paper we present a novel method for discovering knowledge about many of the world's languages by tapping readily available language data posted to the Web. Building upon our work on structural projections across interlinearized text (Xia and Lewis, 2007), we describe a means for automatically discovering a number of computationally salient typological features, such as the existence of particular constituents in a language (*e.g.,* definite or indefinite determiners) or the canonical order of constituents (*e.g.,* sentential word order, order of constituents in noun phrases). This knowledge can then be used for subsequent grammar and tool development work. We demonstrate that given even a very small sample of interlinearized data for a language, it is possible to discover computationally relevant information about the language, and because of the sheer volume and diversity of interlinear text on the Web, it is possible to do so for hundreds to thousands of the world's languages.

*The work described in this document was done while Lewis was faculty at the University of Washington.

2 Background

2.1 Web-Based Interlinear Data as Resource

In linguistics, the practice of presenting language data in interlinear form has a long history, going back at least to the time of the structuralists. Interlinear Glossed Text, or IGT, is often used to present data and analysis on a language that the reader may not know much about, and is frequently included in scholarly linguistic documents. The canonical form, an example of which is shown in (1), consists of three lines: a line for the language in question (often a sentence, which we will refer to here as the *target sentence*), an English gloss line, and an English translation.

(1) Rhoddodd yr athro lyfr i'r bachgen ddoe
gave-3sg the teacher book to-the boy yesterday
"The teacher gave a book to the boy yesterday"
(Bailyn, 2001)

The reader will note that many word forms are shared between the gloss and translation lines, allowing for the alignment between these two lines as an intermediate step in the alignment between the translation and the target. We use this fact to facilitate projections from the parsed English data to the target language, and use the resulting grammars to discover the values of the typological parameters that are the focus of this paper.

We use ODIN, the Online Database of INterlinear text (http://www.csufresno.edu/odin), as our primary source of IGT data. ODIN is the result of an effort to collect and database snippets of IGT contained in scholarly documents posted to the Web (Lewis, 2006). At the time of this writing, ODIN contains 41,581 instances of interlinear data for 944 languages.

2.2 The Structural Projection and CFG Extraction Algorithms

Our algorithm enriches the original IGT examples by building phrase structures over the English data and then projects these onto the target language data via word alignment. The enrichment process has three steps: (1) parse the English translation using an English parser, (2) align the target sentence and the English translation using the gloss line, and (3) project the phrase structures onto the target sentence. The specific details of the projection algorithm are described in (Xia and Lewis, 2007). Given the projected phrase structures on target sentences, we then designed algorithms to extract context-free grammars (CFGs) for each of the languages by reading off the context-free rules from the projected target phrase structure. Identical rules are collapsed, and a frequency of occurrence is associated with each rule. CFGs so generated provide the target grammars we use for work of typological discovery we describe here.

Since the gloss line provides a means of associating the English translation with the target language, the projections from the English translation effectively project "through" the gloss line. Any annotations associated the projected words, such as POS tags, can be associated with words and morphemes on the gloss line during the enrichment process and then can be projected onto the target. These tags are essential for answering some of the typological questions, and are generally not provided by the linguist. This is especially important for associated particular grammatical concepts, such as number or tense, with particular word categories, such as verb and noun.

3 The IGT and English Biases

The choice of the IGT as our source data type presents two causes for concern. First, IGT is typically used by linguists to illustrate linguistically interesting phenomena in a language. A linguist often carefully chooses examples from a language such that they are representative of the phenomena he or she wishes to discuss, and in no way can they be seen as being randomly sampled from a "corpus" of day-to-day usage for the language. It might be argued, then, that a corpus built over IGT suffers from this bias, what we call the *IGT bias*, and results generated from IGT will be somewhat skewed. Second, since we enrich IGT using a method of structural projection from parses made to English translations, the language structures and the grammars extracted from them might suffer from an English-centrism, what we call *English bias*: we cannot assume that all languages will have the same or similar grammatical features or constructions that English has, and by projecting structures from English, we bias the structures we generate to the English source. The degree to which we overcome these biases will demonstrate not only the success of our methodology, but also the viability of a corpus of IGT instances.

4 Experimental Design

4.1 The Typological Parameters

Linguistic typology is the study of the classification of languages, where a typology is an organization of languages by an enumerated list of logically possible types, most often identified by one or more structural features.[1] One of the most well known and well studied typological types, or *parameters*[2], is that of word order, made famous by Joseph Greenberg (Greenberg, 1963). In this seminal work, Greenberg identified six possible orderings of Subjects, Objects, and Verbs in the world's languages, namely, SVO, SOV, VSO, VOS, OSV and OVS, and identified correlations between word order and other constituent orderings, such as the now well known tendency for SVO languages (*e.g.*, English, Spanish) to have prepositional ordering in adpositional phrases and SOV (*e.g.*, Japanese, Korean) to have postpositional.

We take inspiration from Greenberg's work, and that of succeeding typologists (*e.g.*(Comrie, 1989; Croft, 1990)). Using the linguistic typological literature as our base, we identified a set of typological parameters which we felt could have the most relevance to NLP, especially to tasks which might require prototype or structural bootstraps. All of the parameters we identified enumerate various constituent orderings, or the presence or absence of particular constituents. The complete list of typological parameters is shown in table 1. There are two major categories of parameters shown: (1) Constituent order parameters, which are broken down into (a) word order and (b) morpheme order, and (2) constituent existence. For each parameter, we enumerate the list of possible values (what typologists typically call *types*), which is generally a permutation of the possible orderings, constraining the set of possible answers to these values. The value *ndo* is reserved to indicate that a particular language exhibits *no dominant order* for the parameter in question, that is, there is no default or canonical order for the language. The value *nr*, or *not relevant*, indicates that a primary constituent of the parameter does not exist in the language and therefore no possible values for the parameter can exist. A good example of this can be seen for the DT+N parameter: in some languages, definite and indefinite determiners may not exist, therefore making the parameter irrelevant. In the specific case of determiners, we have the Def and Indef parameters, which describe the presence or absence of definite and/or indefinite determiners

[1]See (Croft, 1990) for a thorough discussion of linguistic typology and lists of possible types.

[2]The term *typological parameter* is in line with common usage within the field of linguistic typology.

for any given language. Since the parameters *Def* and *Indef* are strictly existence tests, their possible values are constrained simply to *Yes* or *No*.

4.2 Creating the Gold Standards

The gold standards were created by examining grammars and typological analyses for each language, and in some cases, consulting with native speakers or language experts. A principal target was the *World Atlas of Language Structures*, or WALS (Haspelmath et al., 2005), which contains a typology for hundreds of the world's languages. For each of the parameters shown in Table 1, a WALS # is provided. This was done for the convenience of the reader, and refers to the specific section numbers in WALS that can be consulted for a detailed explanation of the parameter. In some cases, WALS does not discuss a particular parameter we used, in which case a WALS section number is not provided (*i.e.*, it is *N/A*).

5 Finding the Answers

As discussed, a typology consists of a parameter and a list of possible types, essentially the values this parameter may hold. These values are usually not atomic, and can be decomposed into their permuted elements, which themselves are types. For instance, the word order parameter is constrained by the types *SVO*, *SOV*, etc., whose atoms are the types *S* for Subject, *V* for Verb, and *O* for Object. When we talk about the order of words in a language, we are not talking about the order of certain words, such as the constituents *The teacher*, *read*, and *the book* in the sentence *The teacher read the book*, but rather the order of the types that each of these words maps to, *S*, *V*, and *O*. Thus, examining individuals sentences of a language tell us little about the values for the typological parameters if the data is not annotated.

The structural projections built over IGT provide the annotations for specific phrases, words or morphemes in the target language, and, where necessary, the structural relationships between the annotations as expressed in a CFG. There are three broad classes of algorithms for this discovery process, which correspond directly to each of the basic categories of parameters shown in Table 1. For the word order parameters, we use an algorithm that directly examines the linear relationship of the relative types in the CFG. For the DT+N variable, for instance, we look for the relative order of the POS tags DT and N in the NP rules. For the WOrder variable, we look for the relative order NPs and Vs in the S (Sentence) and VP rules. If a language has a dominant rule of S → NP VP, it is highly likely that the language is SVO or SOV, and we can subsequently determine VO or OV by examining the VP rule: VP → V NP indicates VO and VP → NP V indicates OV.

Table 2: Functional Tags in the CFGs

Tag	Meaning	Parameters Affected
NP-SBJ	Subject NP	WOrder, V-OBJ
NP-OBJ	Object NP	WOrder, V-OBJ
NP-POSS	Possessive NP	Poss-N
NP-XOBJ	Oblique Object NP	VP-OBJ
PP-XOBJ	Oblique Object PP	VP-OBJ
DT1	the	DT-N, Def
DT2	a,an	DT-N, Indef
DT3	this, that	Dem-N, Def
DT4	all other determiners	Not used

Determining morpheme order is somewhat simplified in that the CFGs do not have to be consulted, but rather a grammar consisting of possible morpheme orders, which are derived from the tagged constituents on the gloss line. The source of the tags varies: POS tags, for instance, are generally not provided by the linguist, and thus must be projected onto the target line from the English translation. Other tags, such as *case*, *number*, and *tense/aspect* are generally represented by the linguist but with a finer granularity than we need. For example, the linguist will list the specific case, such as NOM for Nominative or ACC for Accusative, rather than just the label "case". We use a table from (Lewis, 2006) that has the top 80 morpheme tags used by linguists to map the specific values to the case, number, and tense/aspect tags that we need.

The existence parameters—in our study constrained to Definite and Indefinite determiners—require us to test the existence of particular POS annotations in the set of relevant CFG rules, and also to examine the specific mappings of words between the gloss and translation lines. For instance, if there are no DT tags in any of the CFG rules for NPs, it is unlikely the language has definite or indefinite determiners. This can specifically be confirmed by checking the transfer rules between *the* and *a* and constituents on the gloss line. If either or both *the* or *a* mostly map to NULL, then either or both may not exist in the language.

6 Experiments

We conducted two experiments to test the feasibility of our methods. For the first experiment, we built a gold standard for each of the typological parameters shown in Table 1 for ten languages, namely Welsh, German, Yaqui, Mandarin Chinese, Hebrew, Hungarian, Icelandic, Japanese, Russian, and Spanish. These languages were chosen for their typological diversity (*e.g.*, word order), for the number of IGT instances available (all had a minimum of fifty instances), and for the fact that some languages were low-density (*e.g.*, Welsh, Yaqui). For the second experiment, we examined the WOrder parameter for 97 languages. The gold standard for this experiment was copied directly from an electronic version of WALS.

Table 1: Computationally Salient Typological parameters (ndo=no dominant order, nr=not relevant)

Label	WALS #	Description	Possible Values
Word Order			
WOrder	330	Order of Words in a sentence	SVO,SOV,VSO,VOS,OVS, OSV,ndo[3]
V+OBJ	342	Order of the Verb, Object and Oblique Object (e.g., PP)	VXO,VOX,OVX,OXV,XVO,XOV,ndo
DT+N	N/A	Order of Nouns and Determiners (*a*, *the*)	DT-N, N-DT, ndo, nr
Dem+N	358	Order of Nouns and Demonstrative Determiners (*this*, *that*)	Dem-N, N-Dem, ndo, nr
JJ+N	354	Order of Adjectives and Nouns	JJ-N, N-JJ, ndo
PRP$+N	N/A	Order of possessive pronouns and nouns	PRP$-N, N-PRP$, ndo, nr
Poss+N	350	Order of Possessive NPs and nouns	NP-Poss, NP-Poss, ndo, nr
P+NP	346	Order of Adpositions and Nouns	P-NP, NP-P, ndo
Morpheme Order			
N+num	138	Order of Nouns and Number Inflections (Sing, Plur)	N-num, num-N, ndo
N+case	210	Order of Nouns and Case Inflections	N-case, case-N, ndo, nr
V+TA	282	Order of Verbs and Tense/Aspect Inflections	V-TA, TA-V, ndo, nr
Existence Tests			
Def	154	Do definite determiners exist?	Yes, No
Indef	158	Do indefinite determiners exist?	Yes, No

Table 3: Experiment 1 Results (Accuracy)

	WOrder	VP +OBJ	DT +N	Dem +N	JJ +N	PRP$ +N	Poss +N	P +NP	N +num	N +case	V +TA	Def	Indef	Avg
basic CFG	0.8	0.5	0.8	0.8	1.0	0.8	0.6	0.9	0.7	0.8	0.8	1.0	0.9	0.800
sum(CFG)	0.8	0.5	0.8	0.8	0.9	0.7	0.6	0.8	0.6	0.8	0.7	1.0	0.9	0.762
CFG w/ func	0.9	0.6	0.8	0.9	1.0	0.8	0.7	0.9	0.7	0.8	0.8	1.0	0.9	0.831
both	0.9	0.6	0.8	0.8	0.9	0.7	0.5	0.8	0.6	0.8	0.7	1.0	0.9	0.769

Since the number of IGT instances varied greatly, from a minimum of 1 (Halkomelem, Hatam, Palauan, Itelmen) to a maximum of 795 (Japanese), as shown in the first column of Table 4, we were able to examine specifically the correlation between the number of instances and our system's performance (at least for this parameter).

6.1 Experiment 1 - Results for 10 Languages, 14 Parameters

As described, the grammars for any given language consist of a CFG and associated frequencies. Our first intuition was that for any given word order parameter, the most frequent ordering, as expressed by the most frequent rule in which it appears, was likely the predominant pattern in the language. Thus, for Hungarian, the order of the DT+N parameter is DT-N since the most frequent rule, namely *NP* $\rightarrow$ *DT N*, occurs much more frequently than the one rule with the opposing order, by a factor of 33 to 1. Our second intuition was based on the assumption that noise could cause an anomalous ordering to appear in the most frequent rule of a targeted type, especially when the number of IGT examples was limited. We hypothesized that "summing" across a set of rules that contained the list of constituents we were interested in might give more accurate results, giving the predominant patterns a chance to reveal themselves in the summation process.

An examination of the types of rules in the CFGs and the parameter values we needed to populate led us to consider enriching the annotations on the English side. For instance, if a CFG contained the rule S $\rightarrow$ NP V, it is impossible for us to tell whether the NP is a subject or an object, a fact that is particularly relevant to the WOrder parameter. We enriched the annotations with functional tags, such as SBJ, OBJ, POSS, etc., which we assigned using heuristics based on our knowledge of English, and which could then be projected onto the target. The downside of such an approach is that it increases the granularity of the grammar rules, which then could weaken the generalizations that might be relevant to particular typological discoveries. However, summing across such rules might alleviate some of this problem. We also divided the English determiners into four groups in order to distinguish their different types, and projected the refined tags onto the target. The full set of functional tags we used are shown in Table 2, with the list of typological parameters that were affected by the inclusion of each.[4] The results for the experiment are shown in Table 3.

[4]It should be noted some "summations" were done to the CFGs in a preprocessing step, thus affecting all subsequent processing. All variants of NN (NN, NNS, NNP) were collapsed into N and all of VB (VB, VBD, VBZ, etc.) into V. Unaligned words and punctuation were also deleted and the affected rules collapsed.

Table 4: Confusion Matrix for the Word Order Types

Word order	# of languages	System Prediction			
		SVO	SOV	VSO	VOS
SVO	46	32	8	0	6
SOV	39	2	33	0	4
VSO	11	2	2	3	4
VOS	1	0	0	0	1

Table 5: Word Order Accuracy for 97 languages

# of IGT instances	Average Accuracy
100+	100%
40-99	99%
10-39	79%
5-9	65%
3-4	44%
1-2	14%

6.2 Experiment 2 Results - Word Order for 97 Languages

The second experiment sought to assign values for the WOrder parameter for 97 languages. For this experiment, a CFG with functional tags was built for each language, and the WOrder algorithm was applied to each language's CFG. The confusion matrix in Table 4 shows the number of correct and incorrect assignments. SVO and SOV were assigned correctly most of the time, whereas VSO produced significant error. This is mostly due to the smaller sample sizes for VSO languages: of the 11 VSO languages in our survey, over half had sample sizes less than 10 IGT instances; of those with instance counts above 70 (two languages), the answer was correct.

6.3 Error Analysis

There are four main types of errors that affected our system's performance:

- Insufficient data – Accuracy of the parameters was affected by the amount of data available. For the WOrder parameter, for instance, the number of instances is a good predictor of the confidence of the value returned. The accuracy of the WOrder parameter drops off geometrically as the number of instances approaches zero, as shown in Table 5. However, even with as few as 4-8 instances, one can accurately predict WOrder's value more than half the time. For other parameters, the absence of crucial constituents (*e.g.*, Poss, PRP$) did not allow us to generate a value.
- Skewed or inaccurate data – Depending on the number of examples and source documents, results could be affected by the *IGT bias*. For instance, although Cantonese (YUH) is a strongly SVO language and ODIN contains 73 IGT instances for the language, our system determined that Cantonese was VOS. This resulted from a large number of skewed examples found in just one paper.
- Projection errors – In many cases, noise was introduced into the CFGs when the word aligner or projction algorithm made mistakes, potentially introducing unaligned constituents. These were subsequently collapsed out of the CFGs. The absent constituents sometimes led to spurious results when the CFGs were later examined.
- Free constituent order – Some languages have freer constituent order than others, making calculation of particular parametric values difficult. For example, Jingulu (JIG) and German (GER) alternate between SVO and SOV. In both cases, our grammars directed us to an order that was opposite our gold standard.

7 Discussion

7.1 Data

In examining Table 5, the reader might question why it is necessary to have 40 or more sentences of parsed language data in order to generalize the word order of a language with a high degree of confidence. After all, anyone could examine just one or two examples of parsed English data to discern that English is SVO, and be nearly certain to be right. There are several factors involved. First, a typological parameter like WOrder is meant to represent a *canonical* characteristic of the language; all languages exhibit varying degrees of flexibility in the ordering of constituents, and discovering the canonical order of constituents requires accumulating enough data for the pattern to emerge. Some languages might require more instances of data to reach a generalization than others precisely because they might have freer word order. English has a more rigid word order than most, and thus would require less data.

Second, the data we are relying on is somewhat skewed, resulting from the IGT bias. We have to collect sufficient amounts of data and from enough sources to counteract any linguist-based biases introduced into the data. It is also the case that not all examples are full sentences. A linguist might be exploring the structure of noun phrases for instance, and not provide full sentences.

Third, we are basing our analyses on projected structures. The word alignment and syntactic projections are not perfect. Consequently, the trees generated, and the rules read off of them, may be incomplete or inaccurate.

7.2 Relevance to NLP

Our efforts described here were inspired by some recent work on low-density languages (Yarowksy and Ngai, 2001; Maxwell and Hughes, 2006; Drabek and Yarowsky, 2006). Until fairly recently, almost all NLP work was done on just a dozen or so languages, with the

vast majority of the world's 6,000 languages being ignored. This is understandable, since in order to do serious NLP work, a certain threshold of corpus size must be achieved. We provide a means for generating small, richly annotated corpora for hundreds of languages using freely available data found on the Web. These corpora can then be used to generate other electronic resources, such as annotated corpora and associated NLP tools.

The recent work of (Haghighi and Klein, 2006) and (Quirk et al., 2005) were also sources of inspiration. In the former case, the authors showed that it is possible to improve the results of grammar induction over raw corpora if one knows just a few facts about the target language. The "prototypes" they describe are very similar to the our constituent order parameters, and we see our work as an incremental step in applying grammar induction to raw corpora for a large number of languages.

Quirk et al 2005 demonstrates the success of using fragments of a target language's grammar, what they call "treelets", to improve performance in phrasal translation. They show that knowing even a little bit about the syntax of the target language can have significant effects on success of phrasal-based MT. Our parameters are in some ways similar to the treelets or grammar fragments built by Quirk and colleagues and thus might be applicable to phrasal-based MT for a larger number of languages.

Although the reader might question the utility of using enriched IGT for discovering the values of typological parameters, since the "one-off" nature of these discoveries might argue for using existing grammars (*e.g.*, WALS) over harvesting and enriching IGT. However, it is important to recognize that the parameters that we specify in this paper are only a sample of the potential parameters that might be recoverable from enriched IGT. Further, because we are effectively building PCFGs for the languages we target, it is possible to provide gradient values for various parameters, such as the degree of word order variability in a language (*e.g.*, SVO 90%, SOV 10%), the potential for which we not explicitly explored in this paper. In addition, IGT exists in one place, namely ODIN, for hundreds of languages, and the examples that are harvested are also readily available for review (not always the case for grammars).

8 Conclusion

We demonstrate a method for discovering interesting and computationally relevant typological features for hundreds of the world's languages automatically using freely available language data posted to the Web. We demonstrate that confidence increases as the number of data points increases, overcoming the IGT and English biases. Inspired by work that uses prototypes and grammar fragments, we see the work we describe here as being quite relevant to the growing body of work on languages whose digital footprint is much smaller than the ten or so majority languages of the world.

References

John Frederick Bailyn. 2001. Inversion, dislocation and optionality in russian. In Gerhild Zybatow, editor, *Current Issues in Formal Slavic Linguistics*.

B. Comrie. 1989. *Language Universals and Linguistic Typology: Syntax and Morphology*. Blackwell, Oxford.

William Croft. 1990. *Typology and Universals*. Cambridge University Press, New York.

Elliott Franco Drabek and David Yarowsky. 2006. Induction of fine-grained part-of-speech taggers via classifier combination and crosslingual projection. In *Proceedings of COLING/ACL2006 Workshop on Frontiers in Linguistically Annotated Corpora*.

Joseph H. Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Language*, pages 73–113. MIT Press, Cambridge, Massachusetts.

Aria Haghighi and Dan Klein. 2006. Protoype-driven sequence models. In *Proceedings of HLT-NAACL*, New York City, NY.

Martin Haspelmath, Matthew S. Dryer, David Gil, and Bernard Comrie. 2005. *The World Atlas of Language Structures*. Oxford University Press, Oxford, England.

William D. Lewis. 2006. ODIN: A Model for Adapting and Enriching Legacy Infrastructure. In *Proceedings of the e-Humanities Workshop*, Amsterdam. Held in cooperation with e-Science 2006: 2nd IEEE International Conference on e-Science and Grid Computing.

Mike Maxwell and Baden Hughes. 2006. Frontiers in linguistic annotation for lower-density languages. In *Proceedings of COLING/ACL2006 Workshop on Frontiers in Linguistically Annotated Corpora*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency tree translation: Syntactically informed phrasal smt. In *Proceedings of ACL 2005*.

Fei Xia and William D. Lewis. 2007. Multilingual structural projection across interlinearized text. In *Proceedings of the North American Association of Computational Linguistics (NAACL) conference*.

David Yarowksy and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of NAACL-2001*, pages 377–404.

Automatic Paraphrasing of Japanese Functional Expressions Using a Hierarchically Organized Dictionary

Suguru Matsuyoshi[†,‡] **Satoshi Sato**[‡]
[†] Graduate School of Informatics, Kyoto University, Japan
[‡] Graduate School of Engineering, Nagoya University, Japan
{s_matuyo,ssato}@nuee.nagoya-u.ac.jp

Abstract

Automatic paraphrasing is a transformation of expressions into semantically equivalent expressions within one language. For generating a wider variety of phrasal paraphrases in Japanese, it is necessary to paraphrase functional expressions as well as content expressions. We propose a method of paraphrasing of Japanese functional expressions using a dictionary with two hierarchies: a morphological hierarchy and a semantic hierarchy. Our system generates appropriate alternative expressions for 79% of source phrases in Japanese in an open test. It also accepts style and readability specifications.

1 Introduction

Automatic paraphrasing is a transformation of expressions into semantically equivalent expressions within one language. It is expected for various applications, such as information retrieval, machine translation and a reading/writing aid.

Automatic paraphrasing of Japanese text has been studied by many researchers after the first international workshop on automatic paraphrasing (Sato and Nakagawa, 2001). Most of them focus on paraphrasing of content words, such as noun phrases and verb phrases. In contrast, paraphrasing of *functional expressions* has less attention. A functional expression is a function word or a multi-word expression that works as a function word. For generating a wider variety of phrasal paraphrases in Japanese, as shown in Fig. 1, it is necessary to paraphrase functional expressions as well as content expressions, because almost all phrases in Japanese include one or more functional expressions. In this paper, we focus on paraphrasing of Japanese functional expressions.

In several applications, such as a reading aid, in paraphrasing of Japanese functional expressions, control of readability of generated text is important, because functional expressions are critical units that determine sentence structures and meanings. In case a reader does not know a functional expression, she fails to understand the sentence meaning. If the functional expression can be paraphrased into an easier one, she may know it and understand the sentence meaning. It is desirable to generate expressions with readability suitable for a reader because easier functional expressions tend to have more than one meaning.

A remarkable characteristic of Japanese functional expressions is that each functional expression has many different variants. Each variant has one of four styles. In paraphrasing of Japanese functional expressions, a paraphrasing system should accept style specification, because consistent use in style is required. For example, the paraphrase (b) in Fig. 1 is not appropriate for a document in normal style because the expression has polite style.

Paraphrasing a functional expression into a semantically equivalent one that satisfies style and readability specifications can be realized as a combination of the following two processes:

1. Transforming a functional expression into another one that is semantically equivalent to it, often with changing readability.

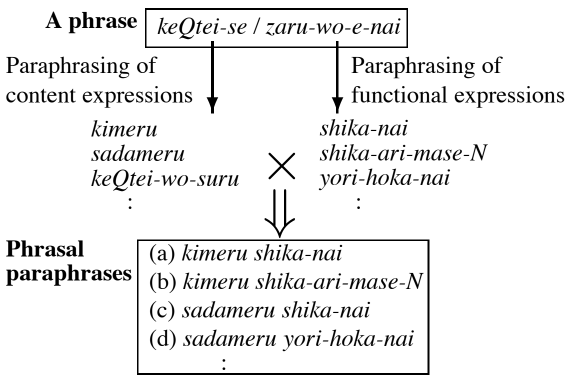


Figure 1: Generation of a wider variety of phrasal paraphrases.

2. Rewriting a functional expression to a variant of it, often with changing style.

We propose a method of paraphrasing of Japanese functional expressions using a dictionary with two hierarchies: a *morphological hierarchy* and a *semantic hierarchy*. The former hierarchy provides a list of all variants specified with style for each functional expression, which is required for the above process 2. The latter hierarchy provides semantic equivalence classes of functional expressions and readability level for each functional expression, which are required for the above process 1.

2 Related Work

A few studies on paraphrasing of Japanese functional expressions have been conducted. In order to implement automatic paraphrasing, some studies (Iida et al., 2001; Tsuchiya et al., 2004) use a set of paraphrasing rules, and others (Tanabe et al., 2001; Shudo et al., 2004) use semantic equivalence classes.

All of these studies do not handle variants in a systematic way. In case a system paraphrases a functional expression f into f', it also should generate all variants of f' in potential. However, any proposed system does not guarantee this requirement. Output selection of variants should be determined according to the given style specification. Any proposed system does not have such selection mechanism.

Controlling readability of generated text is not a central issue in previous studies. An exception is a study by Tsuchiya et al. (Tsuchiya et al., 2004).

Level		Num
L^1	Headword	341
L^2	**Headwords with unique meaning**	**435**
L^3	Derivations	555
L^4	Alternations of function words	774
L^5	Phonetic variations	1,187
L^6	Insertion of particles	1,810
L^7	Conjugation forms	6,870
L^8	Normal or *desu*/*masu* forms	9,722
L^9	Spelling variations	16,801

Table 1: Nine levels of the morphological hierarchy.

Their system paraphrases a functional expression into an easier one. However, it does not accept the readability specification, e.g. for learners of beginner course or intermediate course of Japanese.

3 A Hierarchically Organized Dictionary of Japanese Functional Expressions

3.1 Morphological hierarchy

In order to organize many different variants of functional expressions, we have designed a morphological hierarchy with nine abstraction levels (Matsuyoshi et al., 2006). Table 1 summarizes these nine levels. The number of entries in L^1 (headwords) is 341, and the number of leaf nodes in L^9 (surface forms) is 16,801. For each surface form in the hierarchy, we specified one of four styles (normal, polite, colloquial, and stiff) and connectability (what word can be to the left and right of the expression).

3.2 Semantic hierarchy

There is no available set of semantic equivalence classes of Japanese functional expressions for paraphrasing. Some sets are described in books in linguistics (Morita and Matsuki, 1989; Tomomatsu et al., 1996; Endoh et al., 2003), but these are not for paraphrasing. Others are proposed for paraphrasing in natural language processing (Tanabe et al., 2001; Shudo et al., 2004), but these are not available in public.

For 435 entries in L^2 (headwords with unique meaning) of the morphological hierarchy, from the viewpoint of paraphrasability, we have designed a semantic hierarchy with three levels according to the semantic hierarchy proposed by a book (Morita and Matsuki, 1989). The numbers of classes in the top, middle and bottom levels are 45, 128 and 199, re-

spectively. For each entry in L^2, we specified one of readability levels of A1, A2, B, C, and F according to proficiency level in a book (Foundation and of International Education, Japan, 2002), where A1 is the most basic level and F is the most advanced level.

3.3 Producing all surface forms that satisfy style and readability specifications

For a given surface form of a functional expression, our dictionary can produce all variants of semantically equivalent functional expressions that satisfy style and readability specifications. The procedure is as follows:

1. Find the functional expression in L^2 for a given surface form according to the morphological hierarchy.
2. Obtain functional expressions that are semantically equivalent to the functional expression according to the semantic hierarchy.
3. Exclude the functional expressions that do not satisfy readability specification.
4. Enumerate all variants (surface forms) of the remaining functional expressions according to the morphological hierarchy.
5. Exclude the surface forms that do not satisfy style specification.

4 Formulation of Paraphrasing of Japanese Functional Expressions

As a source expression of paraphrasing, we select a phrase (or *Bunsetsu*) in Japanese because it is a base unit that includes functional expressions. In this paper, we define a phrase as follows. Let c_i be a content word, and f_j a functional expression. Then, a phrase is formulated as the following:

$$\text{Phrase} = c_1 c_2 \cdots c_m f_1 f_2 \cdots f_n, \quad (1)$$

where $c_1 c_2 \cdots c_m$ is the content part of the phrase and $f_1 f_2 \cdots f_n$ is the functional part of it.

Paraphrasing of a functional part of a phrase is performed as a combination of the following five types of paraphrasing:

1→1 Substituting a functional expression with another functional expression ($f \rightarrow f'$).

Paraphrasing type	Num
1→1 only	214 (61%)
1→N (and 1→1)	69 (20%)
N→1 (and 1→1)	18 (5%)
M→N (and 1→1)	8 (2%)
Otherwise	44 (12%)
Sum	353 (100%)

Table 2: Number of paraphrases produced by a native speaker of Japanese.

1→N Substituting a functional expression with a sequence of functional expressions ($f \rightarrow f'_1 f'_2 \cdots f'_N$).

N→1 Substituting a sequence of functional expressions with one functional expression ($f_1 f_2 \cdots f_N \rightarrow f'$).

M→N Substituting a sequence of functional expressions with another sequence of functional expressions ($f_1 f_2 \cdots f_M \rightarrow f'_1 f'_2 \cdots f'_N$).

f⇒c Substituting a functional expression with an expression including one or more content words.

In a preliminary experiment, we investigated which type of the above a native speaker of Japanese tended to use in paraphrasing a functional part. Table 2 shows the classification result of 353 paraphrases produced by the subject for 238 source phrases.[1] From this table, it was found out that paraphrasing of "1→1" type was major in that it was used for producing 61% of paraphrases.

Because of dominance of paraphrasing of "1→1" type, we construct a system that paraphrases Japanese functional expressions in a phrase by substituting a functional expression with a semantically equivalent expression. This system paraphrases a phrase defined as the form in Eq. (1) into the following form:

$$\text{Alternative} = c_1 c_2 \cdots c_{m-1} c'_m w f'_1 f'_2 \cdots f'_n,$$

where c'_m is c_m or a conjugation form of c_m, f'_j is a functional expression that is semantically equivalent to f_j, and w is a null string or a function word that is inserted for connecting f'_1 to c'_m properly.

[1] These source phrases are the same ones that we use in a closed test in section 6.

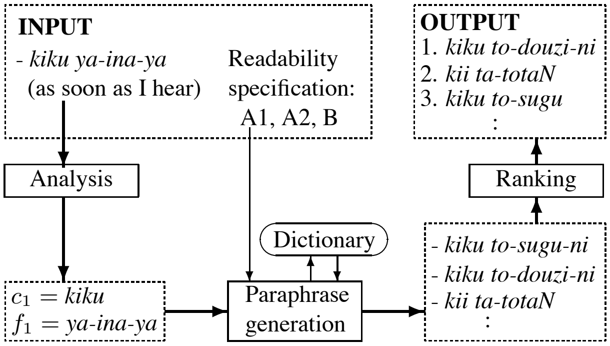


Figure 2: Overview of our system.

The combination of simple substitution of a functional expression and insertion of a function word covers 22% (15/69) of the paraphrases by paraphrasing of "1→N (and 1→1)" type in Table 2. Therefore, our system theoretically covers 65% (229/353) of the paraphrases in Table 2.

5 System

We have implemented a system that paraphrases Japanese functional expressions using a hierarchically organized dictionary, by substituting a functional expression with another functional expression that is semantically equivalent to it. The system accepts a phrase in Japanese and generates a list of ranked alternative expressions for it. The system also accepts style and readability specifications.

Fig. 2 shows an overview of our system. This system consists of three modules: analysis, paraphrase generation, and ranking.

5.1 Analysis

Some methods have been proposed for detecting Japanese functional expressions based on a set of detection rules (Tsuchiya and Sato, 2003) and machine learning (Uchimoto et al., 2003; Tsuchiya et al., 2006). However, because these methods detect only a limited number of functional expressions (and their variants), we cannot apply them to the analysis of a phrase. Another method is to add a list of about 17,000 surface forms of functional expressions to a dictionary of an existing morphological analyzer and determine connecting costs based on machine learning. However, it is infeasible because there is no large corpus in which all of these surface forms have been tagged.

Instead of these methods, we use a different method of decomposing a given phrase into a sequence of content words and functional expressions. Our method uses two analyzers.

We constructed a functional-part analyzer (FPA). This is implemented using a morphological analyzer MeCab[2] with a special dictionary containing only functional expressions. FPA can decompose a functional part (string) into a sequence of functional expressions, but fails to decompose a string when the string includes one or more content words. In order to extract a functional part from a given string, we use original MeCab.

First, original MeCab decomposes a given string into a sequence of morphemes $m_1 m_2 \cdots m_k$. Next, we suppose that m_1 is a content part and $m_2 m_3 \cdots m_k$ is a functional part. If FPA can decompose $m_2 m_3 \cdots m_k$ into a sequence of functional expressions $f_1 f_2 \cdots f_n$, then we obtain $c_1 f_1 f_2 \cdots f_n$ as shown in Eq. (1) as an analyzed result, where $c_1 = m_1$. Otherwise, we suppose that $m_1 m_2$ is a content part and $m_3 m_4 \cdots m_k$ is a functional part. If FPA can decompose $m_3 m_4 \cdots m_k$ into a sequence of functional expressions $f_1 f_2 \cdots f_n$, then we obtain $c_1 c_2 f_1 f_2 \cdots f_n$ as an analyzed result, where $c_1 = m_1$ and $c_2 = m_2$. This procedure is continued until FPA succeeds in decomposition.

5.2 Paraphrase generation

This module accepts an analyzed result $c_1 c_2 \cdots c_m f_1 f_2 \cdots f_n$ and generates a list of alternative expressions for it.

First, the module obtains a surface form f'_1 that is semantically equivalent to f_1 from the dictionary in section 3. Next, it constructs $c_1 c_2 \cdots c_{m-1} c'_m w f'_1$ by connecting f'_1 to $c_1 c_2 \cdots c_m$ by the method described in section 4. Then, it obtains a surface form f'_2 that is semantically equivalent to f_2 and constructs $c_1 c_2 \cdots c_{m-1} c'_m w f'_1 f'_2$ in similar fashion. This process proceeds analogously, and finally, the module constructs $c_1 c_2 \cdots c_{m-1} c'_m w f'_1 f'_2 \cdots f'_n$ as an alternative expression.

Because in practice the module obtains more than one surface form that is semantically equivalent to

[2] http://mecab.sourceforge.net/

	Top 1	Top 1 to 2	Top 1 to 3	Top 1 to 4	Top 1 to 5
Closed	177 (74%)	197 (83%)	**210 (88%)**	213 (90%)	213 (90%)
Closed (Perfect analysis)	196 (82%)	211 (89%)	**219 (92%)**	221 (93%)	221 (93%)
Open	393 (63%)	461 (73%)	**496 (79%)**	500 (80%)	501 (80%)
Open (Perfect analysis)	453 (72%)	508 (81%)	**531 (85%)**	534 (85%)	534 (85%)

Table 3: Evaluation of paraphrases generated by the paraphrasing system

f_j by the method described in subsection 3.3, it generates more than one alternative expression by considering all possible combinations of these surface forms and excluding candidates that include two adjacent components that cannot be connected properly.

If the module generates no alternative expression, it uses the semantic equivalence classes in the upper level reluctantly.

5.3 Ranking

Because a functional expression seems to be more standard and common as it appears more frequently in newspaper corpus, we use frequencies of functional expressions (strings) in newspaper corpus in order to rank alternative expressions. We define a scoring function as the product of frequencies of functional expressions in a phrase.

6 Evaluation

We evaluate paraphrases generated by our paraphrasing system for validating our semantic equivalence classes, because the dictionary that the system uses guarantees by the method described in subsection 3.3 that the system can generate all variants of a functional expression and accept style and readability specifications.

6.1 Methodology

We evaluated paraphrases generated by our paraphrasing system from the viewpoint of an application to a writing aid, where a paraphrasing system is expected to output a few good alternative expressions for a source phrase.

We evaluated the top 5 alternative expressions generated by the system for a source phrase by classifying them into the following three classes:

Good Good alternative expression for the source phrase.

Intermediate Expression that keeps the meaning roughly that the source phrase has.

Bad Inappropriate expression.

Then, we counted source phrases for which at least one of the alternative expressions of the top 1 to n was judged as "Good". One of the authors performed the judgment according to books (Morita and Matsuki, 1989; Endoh et al., 2003).

As a closed test set, we used 238 example phrases for 140 functional expressions extracted from a book (Foundation and of International Education, Japan, 2002), which we had used for development of our semantic equivalence classes. As an open test set, we used 628 example phrases for 184 functional expressions extracted from a book (Tomomatsu et al., 1996). We used the Mainichi newspaper text corpus (1991-2005, about 21 million sentences, about 1.5 gigabytes) for ranking alternative expressions.

6.2 Results

Table 3 shows the results. The rows with "Perfect analysis" in the table show the results in analyzing source phrases by hand. Because the values in every row of the table are nearly saturated in "Top 1 to 3", we discuss the results of the top 1 to 3 hereafter.

Our system generated appropriate alternative expressions for 88% (210/238) and 79% (496/628) of source phrases in the closed and the open test sets, respectively. We think that this performance is high enough.

We analyzed the errors made by the system. In the closed and the open tests, it was found out that paraphrasing of "1→1" type could not generate alternative expressions for 7% (16/238) and 7% (41/628) of source phrases, respectively. These values define the upper limit of our system.

In the closed and the open tests, it was found out that the system failed to analyze 3% (8/238) and 3% (21/628) of source phrases, respectively, and that

ambiguity in meaning caused inappropriate candidates to be ranked higher for 1% (2/238) and 4% (23/628) of source phrases, respectively. The rows with "Perfect analysis" in Table 3 show that almost all of these problems are solved in analyzing source phrases by hand. Improvement of the analysis module can solve these problems.

In the open test, insufficiency of semantic equivalence classes and too rigid connectability caused only 3% (19/628) and 3% (16/628) of source phrases to have no good candidates, respectively. The smallness of the former value validates our semantic equivalence classes.

The remaining errors were due to low frequencies of good alternatives in newspaper corpus.

7 Conclusion and Future Work

We proposed a method of paraphrasing Japanese functional expressions using a dictionary with two hierarchies. Our system can generate all variants of a functional expression and accept style and readability specifications. The system generated appropriate alternative expressions for 79% of source phrases in an open test.

Tanabe et al. have proposed paraphrasing rules of "1→N", "N→1", and "M→N" types (Tanabe et al., 2001). For generating a wider variety of phrasal paraphrases, future work is to incorporate these rules into our system and to combine several methods of paraphrasing of content expressions with our method.

References

Orie Endoh, Kenji Kobayashi, Akiko Mitsui, Shinjiro Muraki, and Yasushi Yoshizawa, editors. 2003. *A Dictionary of Synonyms in Japanese (New Edition)*. Shogakukan. (in Japanese).

The Japan Foundation and Association of International Education, Japan, editors. 2002. *Japanese Language Proficiency Test: Test Content Specifications (Revised Edition)*. Bonjinsha. (in Japanese).

Ryu Iida, Yasuhiro Tokunaga, Kentaro Inui, and Junji Etoh. 2001. Exploration of clause-structural and function-expressional paraphrasing using KURA. In *Proceedings of the 63rd National Convention of Information Processing Society of Japan*, volume 2, pages 5–6. (in Japanese).

Suguru Matsuyoshi, Satoshi Sato, and Takehito Utsuro. 2006. Compilation of a dictionary of Japanese functional expressions with hierarchical organization. In *Proceedings of the 21st International Conference on Computer Processing of Oriental Languages (ICCPOL), Lecture Notes in Computer Science*, volume 4285, pages 395–402. Springer.

Yoshiyuki Morita and Masae Matsuki. 1989. Nihongo Hyougen Bunkei*, volume 5 of* NAFL Sensho *(Expression Patterns in Japanese)*. ALC Press Inc. (in Japanese).

Satoshi Sato and Hiroshi Nakagawa, editors. 2001. *Automatic Paraphrasing: Theories and Applications*, The 6th Natural Language Processing Pacific Rim Symposium (NLPRS) Post-Conference Workshop.

Kosho Shudo, Toshifumi Tanabe, Masahito Takahashi, and Kenji Yoshimura. 2004. MWEs as non-propositional content indicators. In *Proceedings of the 2nd ACL Workshop on Multiword Expressions: Integrating Processing (MWE-2004)*, pages 32–39.

Toshifumi Tanabe, Kenji Yoshimura, and Kosho Shudo. 2001. Modality expressions in Japanese and their automatic paraphrasing. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLPRS)*, pages 507–512.

Etsuko Tomomatsu, Jun Miyamoto, and Masako Wakuri. 1996. *500 Essential Japanese Expressions: A Guide to Correct Usage of Key Sentence Patterns*. ALC Press Inc. (in Japanese).

Masatoshi Tsuchiya and Satoshi Sato. 2003. Automatic detection of grammar elements that decrease readability. In *Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, pages 189–192.

Masatoshi Tsuchiya, Satoshi Sato, and Takehito Utsuro. 2004. Automatic generation of paraphrasing rules from a collection of pairs of equivalent sentences including functional expressions. In *Proceedings of the 10th Annual Meeting of the Association for Natural Language Processing*, pages 492–495. (in Japanese).

Masatoshi Tsuchiya, Takao Shime, Toshihiro Takagi, Takehito Utsuro, Kiyotaka Uchimoto, Suguru Matsuyoshi, Satoshi Sato, and Seiichi Nakagawa. 2006. Chunking Japanese compound functional expressions by machine learning. In *Proceedings of the workshop on Multi-word-expressions in a multilingual context, EACL 2006 Workshop*, pages 25–32.

Kiyotaka Uchimoto, Chikashi Nobata, Atsushi Yamada, Satoshi Sekine, and Hitoshi Isahara. 2003. Morphological analysis of a large spontaneous speech corpus in Japanese. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 479–488.

Generation of Referring Expression Using Prefix Tree Structure

Sibabrata Paladhi
Department of Computer Sc. & Engg.
Jadavpur University, India
sibabrata_paladhi@yahoo.com

Sivaji Bandyopadhyay
Department of Computer Sc. & Engg.
Jadavpur University, India
sivaji_cse_ju@yahoo.com

Abstract

This paper presents a Prefix Tree (Trie) based model for Generation of Referring Expression (GRE). The existing algorithms in GRE lie in two extremities. Incremental algorithm is simple and speedy but less expressive in nature whereas others are complex and exhaustive but more expressive in nature. Our prefix tree based model not only incorporates all relevant features of GRE (like describing set, generating Boolean and context sensitive description etc.) but also try to attain simplicity and speed properties of Incremental algorithm. Thus this model provides a simple and linguistically rich approach to GRE.

1 Introduction

Generation of referring expression (GRE) is an important task in the field of Natural Language Generation (NLG) systems (Reiter and Dale, 1995). The task of any GRE algorithm is to find a combination of properties that allow the audience to identify an object (target object) from a set of objects (domain or environment). The properties should satisfy the target object and dissatisfy all other objects in the domain. We sometimes call it distinguishing description because it helps us to distinguish the target from potential distractors, called contrast set. When we generate any natural language text in a particular domain, it has been observed that the text is centered on certain objects for that domain. When we give introductory description of any object, we generally give full length description (e.g. “The large black hairy dog”). But the later references to that object tend to be shorter and only support referential communication goal of distinguishing the target from other objects. For example the expression “The black dog” suffices if the other dogs in the environment are all non black. Grice, an eminent philosopher of language, has stressed on brevity of referential communication to avoid conversational implicature. Dale (1992) developed Full Brevity algorithm based on this observation. It always generates shortest possible referring description to identify an object. But Reiter and Dale (1995) later proved that Full Brevity requirement is an NP-Hard task, thus computationally intractable and offered an alternative polynomial time Incremental Algorithm. This algorithm adds properties in a predetermined order, based on the observation that human speakers and audiences prefer certain kinds of properties when describing an object in a domain (Krahmer et al. 2003). The Incremental Algorithm is accepted as state of the art algorithm in NLG domain. Later many refinements (like Boolean description and set representation (Deemter 2002), context sensitivity (Krahmer et al 2002) etc) have been incorporated into this algorithm. Several approaches have also been made to propose an alternative algorithmic framework to this problem like graph-based (Krahmer et al. 2003), conceptual graph based (Croitoru and Deemter 2007) etc that also handle the above refinements. In this paper we propose a new Prefix Tree (Trie) based framework for modeling GRE problems. Trie is an ordered tree data structure which allows the organization of prefixes in such a way that the branching at each level is guided by the parts of prefixes. There are several advantages of this approach: 1) Trie data structure has already been extensively used in many domains where search is the key operation. 2) The structure is scalable and various optimized algorithms are there for time, space optimizations.

In this paper it is shown how scenes can be represented using a Trie (section 2) and how description generation can be formalized as a search problem (section 3). In section 4 the algorithm is explained using an example scene. In section 5, the basic algorithm is extended to take care of different scenarios. The algorithm is analyzed for time com-

plexity in section 6 and conclusion is drawn in section 7.

2 Modeling GRE Using Trie Structure

In this section, it is shown how a scene can be represented using a trie data structure. The scheme is based on Incremental algorithm (Reiter and Dale 1995) and incorporates the attractive properties (e.g. speed, simplicity etc) of that algorithm. Later it is extended to take care of different refinements (like relational, boolean description etc) that could not be handled by Incremental algorithm. Reiter and Dale (1995) pointed out the notion of '**PreferredAttributes**' (e.g. Type, Size, Color etc) which is a sequence of attributes of an object that human speakers generally use to identify that object from the contrast set. We assume that the initial description of an entity is following this sequence (e.g. "The large black dog") then the later references will be some subset of initial description (like "The dog" or "The large dog") which is defined as the prefix of the initial description. So, we have to search for a prefix of the initial full length description so that it is adequate to distinguish the target object. Following the Incremental version we will add properties one by one from the '**PreferredAttributes'** list. In our model, the root consists of all entities in the domain and has empty description. Then at each level, branching is made based on different values of corresponding preferred attribute. The outgoing edge is labeled with that value. For example, at the first level, branching is made based on different values of 'Type' attribute like 'Dog', 'Cat', 'Poodle' etc. A node in Trie will contain only those objects which have the property(s) expressed by the edges, constituting the path from root to that node. After construction of the Trie structure for a given domain in this way, referring expression generation problem for an object **r** is reduced to search the tree for a node which consists of **r** and no other object. Description for **r** can be found from the search path itself as we have said earlier. Now we will introduce some notations that we will use to describe the actual algorithm. Let **D** be the Domain, **r** be the target object and **P** be the '**PreferredAttributes'** List. $[\![N_i]\!] = \{d \mid d \in D$ and d is stored at node $N_i\}$ where N_i is an i-th level node. Obviously $[\![N_o]\!] = D$ since N_o is root node. $E(N_i, N^k_{i+1})$ is an edge between parent node N_i and N^k_{i+1}, k-th child of that node (considering an enumeration among children nodes). Since every edges in Trie are labeled, thus $\{E\} \subseteq \{N\} \times L \times \{N\}$, where {E} and {N} are set of all edges and nodes respectively in the tree and L is the set of attribute values. Let $Val(E(N_i, N^k_{i+1}))$ denotes the label or value of the edge and $[\![Val(E(N_i, N^k_{i+1}))]\!] = \{d \mid d \in D$ and d is satisfied by the edge value$\}$ i.e. the set contains those objects who have this property. We define $[\![N^k_{i+1}]\!] = [\![N_i]\!] \cap [\![Val(E(N_i, N^k_{i+1}))]\!]$ where N_i and N^k_{i+1} are parent and child node respectively. Similarly $[\![N^k_i]\!] = [\![N_{i-1}]\!] \cap [\![Val(E(N_{i-1}, N^k_i))]\!]$. Ultimately, we can say that $\forall i\ [\![N_i]\!] = [\![N_o]\!] \cap [\![Val(E(N_o,N_1))]\!] \cap \ldots\ldots \cap [\![Val(E(N_{i-1},N_i))]\!]$. Since our construction is basically a tree, each node is reachable from root and there exists a unique path from root to that node. So, for each node in the tree we will get some description. We will formulate referring expression construction as search in the constructed tree for the node $_{\mathbf{min(k)}}\{\mathbf{N_k}\}$ such that $[\![N_k]\!] = \{\mathbf{r}\}$. If N_k is leaf node then description of **r** will be same with the full description but if it is an intermediate node then description is some proper prefix of initial description. But the point is that, in both cases the later reference is prefix of initial one (as both "ab" and "abc" are prefixes of "abc").

3 Basic Algorithm

Based on above discussions, algorithms are developed for construction of Trie from the domain and generation of reference description for any object in that domain. The Trie construction algorithm **ConstructTrie(D,P,T)** is shown in figure 1, Referring expression generation algorithm **MakeRefExpr(r,p,T,L) is** shown in figure 2, where T is a node pointer and p is pointer to parent of that node. Our algorithm **MakeRefExpr** returns set of attribute-values L to identify **r** in the domain. As discussed earlier, it is basically a node searching algorithm. In course of searching, if it is found that an intermediate node **N** doesn't have **r** i.e. $\mathbf{r} \notin [\![\mathbf{N}]\!]$ then our search will not move forward through the subtree rooted at **N**. Our search will proceed through next level iff $\mathbf{r} \in [\![\mathbf{N}]\!]$**.** For a node $\mathbf{N_k}$, if we get $[\![\mathbf{N_k}]\!] = \{\mathbf{r}\}$ then we have succeeded and our algorithm will return **L**, set of descriptions for that node. If there is no distinguishing description exists for **r**, then $\varnothing$ (null) will be returned**.** We

would like to point out that our algorithm will find out only one description that exists at the minimum level of the tree. Moreover, a description is added to **L** only if it is distinguishing i.e. the connecting edge must remove some contrasting object(s). Thus, the child node should contain less number of objects than that of parent node. In this case, cardinality of parent $\mathbf{N_i}$ **(**Card**(**$\mathbf{N_i}$**))** will be greater than that of child **(**Card($\mathbf{N_{i+1}}$)**)**. This condition is included in our algorithm and if (Card (P→N)) > Card (T→N) holds then only the value is added

```
ConstructTrie(D, P, T) {
 If (D = ∅  ∨  P = ∅ )
 Then Stop
 Else
   Create a node N at T
   Set ⟦ N ⟧ = D
   Extract front attribute Ai from list P
   P′ =  P – { Ai }
   For each value Vj  of  attribute  Ai  do
       Create Edge Ej with label Vj as T→Nextj
       Dj′ = D ∩   ⟦ Val(Ej)  ⟧
       ConstructTrie(Dj′ , P′, T→Nextj)
   End For
  End If
}
```

Figure 1. Prefix Tree Generation Algorithm

```
MakeRefExpr(r, P, T, L) {
   If ( r ∉ ⟦ T→N ⟧ )
        Then  L ← ∅
         Return L
    Else If ({r} = ⟦ T→N ⟧)
        L = L ∪ Val(P→Ej )
        Return L
   Else If (isLeaf (T)  ∧ {r} ⊂ ⟦ N ⟧)
         Then L ← ∅
         Return L
   Else {
       If (Card(P→N) > Card (T→N))
         Then L = L ∪ Val(P→Ej )
       P = T
       For each outgoing edge T→ Nextj (Ej)  do
         L′ = MakeRefExpr(r, P,T→ Childj, L)
            If (L′ ≠ ∅ )
            Then Return L′
      } }
```

Figure 2. Expression Generation Algorithm

P->N and T->N respectively represents parent and child node. After finding a distinguishing description for **r,** search will neither move further down the tree nor explore the remaining branches of the current node. Search will explore the next branch only if the search in current branch returned NULL description i.e. when L′ = ∅ in the algorithm. If we reach a leaf node and that contains **r** along with other objects then it is not possible to distinguish **r**'. In that case, the algorithm returns NULL indicating that no description exists at all. It has been later shown that some distinguishing description may still exist and the algorithm will be modified to find that. It should be mentioned that once the prefix tree is constructed offline, it can be used repetitively to find description for any object in the domain throughout the text generation phase. Our **MakeRefExpr()** algorithm is very simple and it doesn't employ any set theoretic operation, which is a non trivial task, to find current contrast set at every steps of algorithm. In existing algorithms, computing referential description for every object require computing similar things (like finding current contrast set, ruled out objects) again and again. And it has to be repeated every time the object is referred. It is not possible to generate description once, store it and use it later because of the fact that domain may also change in course of time (Krahmer, 2002). That's why every time we want to refer to 'r', such rigorous set operations need to be computed. But in our prefix tree structure, once the tree is constructed, it is very easy to find description for that object using simple tree search function. It is also very easy to add/delete objects to/from domain. We have to follow just the initial properties of that object to find the proper branching at each level, followed by addition /deletion of that object to /from relevant nodes, which is essentially a search operation. The disadvantage of our algorithm is that space complexity is high but it can be tackled using bit Vector representation of individual nodes of the prefix tree. Besides, several methods are there for compressing Trie structure. But these optimization techniques are beyond the scope of our current discussion.

4 Formalizing A Scene using Prefix Tree

Consider an example scene in figure 3, from [Krahmer 2002]. In this scene, there is a finite domain of entities D. Let D = {d1, d2, d3, d4}, P = {Type, Size, Color} and values are Type = {dog, cat}; Size = {small, large}; Color = {black, white}. A scene is usually represented as a database (or

knowledge base) listing the properties of each element in D. Thus:

d_1 : ⟨ Type : dog ⟩ , ⟨ Size : small ⟩ , ⟨ Color: white ⟩
d_2 : ⟨ Type : dog ⟩ , ⟨ Size : large ⟩ , ⟨ Color: white ⟩
d_3 : ⟨ Type : dog ⟩ , ⟨ Size : large ⟩ , ⟨ Color: black ⟩
d_4: ⟨ Type : cat ⟩ , ⟨ Size: small ⟩ , ⟨ Color: white ⟩

Now it will be shown how our **MakeRefExpr()** algorithm will find a description for a target object **r**. Let **r** = {d_1}. In the first phase, starting from root, edge labeled D is traversed. Since d_1 exists in the node and D discards some objects (d_4), D is distinguishing description and it is added to L. In the next phase the node connected by the edge labeled L does not contain d_1 so search will not proceed further. Rather the node connected by the edge labeled S contains d_1. Since, d_1 is the only object, then we are done and the referring expression is "The small dog". But for d_2, we have to search upto the leaf node which generates the description "The large white dog".

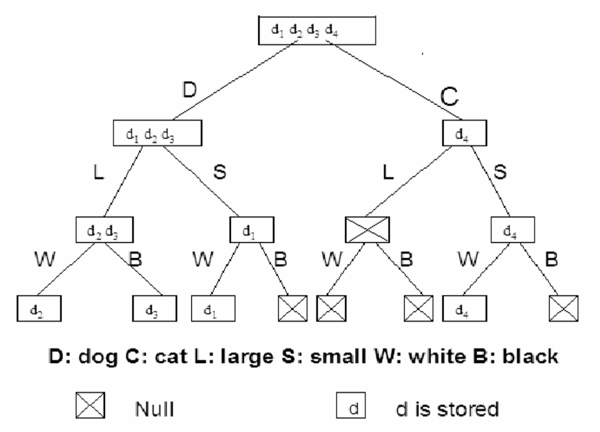


Figure 3. Scene Representation

5 Extension of Basic Algorithm

5.1 Specifying Overlapping Values

Deemter (2002) has shown incompleteness of Incremental algorithm in case of overlapping values. Due to vagueness of properties, sometimes it is hard to classify an object in a particular class. Consider the example scene D = {a,b,c,d} Color: {Red(a,b); Orange(a,c,d)} Size: {Large(a,b); Small(c,d)}. In this case **a** can not be properly classified by Color type. Incremental algorithm always select Red(a,b) at first phase, since it rules out maximum distractors and returns failure because it can't distinguish **a** from **b** at second phase. Deemter(2002) suggested inclusion of all overlapping values that are true of target while also removing some distractors. So, referring expression for **a** is "The red orange desk". But it fails to obey Gricean maxims of conversational implicature. We consider the failure as 'Early Decision' problem and defer the decision making in our model. We keep in our mind the fact that human beings seldom take instantaneous decision. Rather they consider all opportunities in parallel and take decision in the favor of the best one at later point of time. Since, our algorithm searches in parallel through all promising branches until some description is found; it mimics the capabilities of human mind to consider in parallel. Our algorithm will generate "The large orange desk" which will help audiences to better identify the desk. The execution sequence is shown in figure 4.

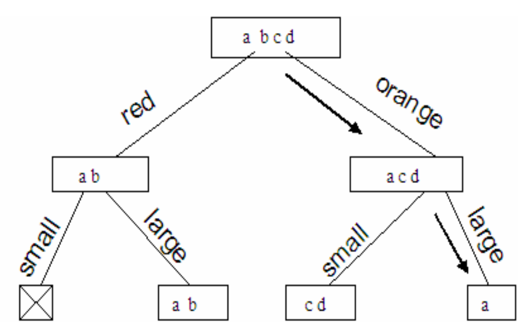


Figure 4. Dealing with overlapping values

5.2 Describing Set of Objects

Generation of referring description for a set of objects is very important in NLG. Deemter's (2002) suggestion can be easily incorporated into our framework. We will represent target **r** as set of objects. Now our algorithm will try to find a node in the tree which only consists of all objects in the set **r**. In this way, we can find a distinguishing description for any set, for which description exists. In figure 3, the description for the set {d_2,d_3} is "The large dogs". Thus, our basic algorithm is able to describe set of objects. In case of set like {d_2, d_3, d_4} where there is no separate node consisting all the object, we need to partition the set and find description for individual set. In our case the possible partitions are {d_2, d_3} and {d_4} for which separate nodes exist.

5.3 Boolean Descriptions

Deemter (2002) shown that Incremental algorithm is only intersectively complete. But he argues that other Boolean combination of properties can be used to generate description for an object. Consider the example from (Deemter, 2002). Let D = {a, b, c, d, e} Type: {Dog(a,b,c,d,e); Poodle(a,b)} Color: {Black(a,b,c); White(d,e)} and r = {c}. In this scenario Incremental algorithm is not able to individuate any of the animals. However a description for c exists, "The black dog that is not a poodle". Since $\{c\} = [[\text{Black}]] \cap [[\neg \text{Poodle}]]$. Deemter (2002) has modified the Incremental algorithm by adding negative values for each attribute. Now we will show that our basic algorithm can be modified to take care of this situation. In our basic algorithm **ConstructTrie()**, we add branches at each level for negative values also. In this case our simple routine **MakeRefExpr() is** able to find boolean description while remaining as close as to Incremental algorithm. In figure 5, we show part of the trie structure, which is generated for the above scene. The dashed arrows show the alternative search paths for node containing {c}.

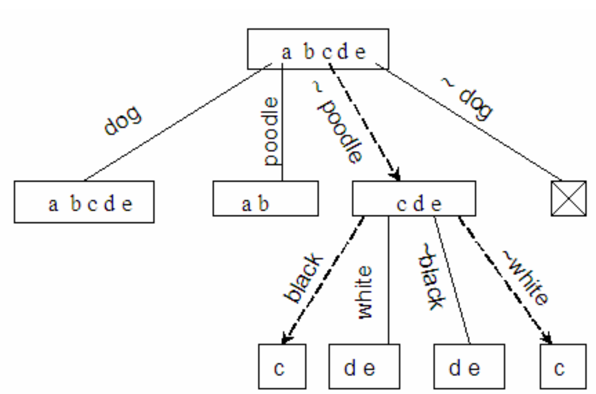


Figure 5. Trie structure (Partial) incorporating negation of properties

For referring objects using disjunction of properties we have do same thing as negations. We have to extend our prefix tree structure by adding extra edges at different levels for making implicit information explicit as described in [Krahmer 2002].

5.4 Incorporating Context Sensitivity

Krahmer and Theune [2002] have added the notion of context sensitivity into GRE. Earlier algorithms assumed that all objects in environment are equally salient. Krahmer and Theune refined the idea by assigning some degree of salience to each object. They proposed that during referring any object, the object needs to be distinguished only from those objects which are more salient (having higher salience weight). An object that has been mentioned recently, is linguistically more salient than other objects and can be described using fewer properties ("The dog" instead of "The large black hairy dog"). They introduced the concept of centering theory, hierarchical focus constraints in the field of NLG and devised a constant function mapping **sw:** $\mathbf{D} \rightarrow \mathbb{N}$, where **sw** is salience weight function, **D** is domain and $\mathbb{N}$ is set of natural numbers. We can incorporate this idea into our model easily. In each node of the prefix tree we keep a field 'salience weight' (sw) for each of the object stored in that node in the form (d_i, sw_i). During describing an object if we find a node that is containing **r** where it is the most salient then we need not traverse higher depth of the tree. So, we have to modify **MakeRefExpr()** algorithm by adding more conditions. If the current node is **N** and both **1)** $\mathbf{r} \in [\![\mathbf{N}]\!]$ **and 2)** $\forall \mathbf{d} \in [\![\mathbf{N}]\!]\ (\mathbf{d} \neq \mathbf{r} \rightarrow \mathbf{sw(d)} < \mathbf{sw(r)})$ hold then **r** is the most salient and the edges constituting the path from root to **N** represents distinguishing description for **r**. In figure 6, **a** is most salient dog and referred to as "The dog" whereas **b** is referred to as "The small dog".

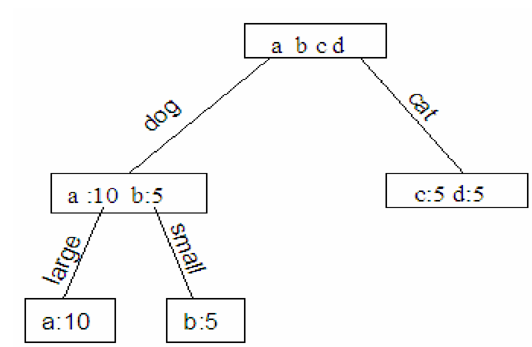


Figure 6: Trie structure (Partial) representing Context Sensitivity

5.5 Relational Descriptions

Relational descriptions are used to single out an object with reference to other one. For example "The cup on the table" is used to distinguish a cup from other cups which are not on the table. Dale and Haddock (1991) first offer the idea of rela-

tional description and extend Full Brevity algorithm to incorporate this idea. Later Krahmer et al. (2003) Graph based framework for generating relational description. We follow Krahmer (2002) and denote relations as Spatial: {In(a,b); Left_of(c,d)} etc. Then we treat 'Spatial' as another attribute and consider 'In', 'Left_of' as different values for that attribute. In this way, our basic algorithm itself is capable of handling relational descriptions. The only modification that we add that when a relation R is included, the **MakeRefExpr()** should be called again for the relatum. Thus, if $Val(E(N_i, N^k_{i+1}))$ expresses a relation of **r** with **r′** then we have to call **MakeRefExpr (r′,p,T,L)** again to find description for **r′.**

5.6 Modeling Full Brevity

In this section, we will show that our prefix tree structure can be so modified that it can generate shortest possible description which is requirement of Full Brevity (Dale, 1992). Consider a scene where a domain is identified by set of n attributes $\{A_1, A_2 \ldots A_n\}$. We can generate n! number of different permutations of A_i's $\forall\ i \in [1,n]$. We consider each permutation as different **PreferredAttributes** list P_k and generate all possible prefix trees T_k for each P_k $\forall\ k \in [1,n!]$ for same domain D. Now, we connect roots of all trees with a common dummy root node with edges having empty description (ε). Now, if we search the branches of new combined tree in parallel, it's obvious that we can always find the target node at lowest possible level. Thus we can generate shortest length description using our algorithm.

6 Complexity of The Algorithm

Let the domain entities are identified by **a** number of attributes and each attribute has on the average **k** number of different values. So, our **ConstructTrie()** algorithm takes $O(k^a)$ time. Now we will consider different cases for analyzing the time complexity of our MakeRefExpr() algorithm.

1) In case of non overlapping properties, our search tree will be pruned at each level by a factor of k. Thus the time complexity will be $O(\log_k(k^a)) = O(a)$ which is linear.

2) In case of overlapping properties, we have to search whole tree in worst case (although in average cases also there will be large pruning, as found from test cases) which will take $O(k^a)$ time.

3) In case of achieving full brevity requirement, both time and space complexity will be exponential as in the original algorithm by Dale (1992).

7 Conclusions

In this paper, we present a new Prefix tree (Trie) based approach for modeling GRE problems. We construct the trie in such a way that a node at a particular level consists of only those objects which are satisfied by values of the edges, constituting the path from root to that node. We formulate description generation as a search problem. So, when we reach the target node, the attribute values corresponding to the edges in the path automatically form the distinguishing description. Different scenarios of GRE problems like representation of set, boolean descriptions etc. is taken care of in this paper. We have shown that in simple non overlapping scenarios, our algorithm will find distinguishing description in linear time.

8 References

E. Krahmer and M. Theune. 2002. Efficient Context Sensitive Generation of Referring Expressions. *CSLI Publ, Stanford* : 223 – 264

E. Krahmer, S. van Erk and A. Verlag. 2003. Graph based Generation of Referring Expressions *Computational Linguistics,* 29(1): 53-72

H. Horacek. 2004. On Referring to Set of Objects Naturally. *Proceedings of Third INLG, Brokenhurst, U.K: 70-79*

M. Croitoru and van Deemter. 2007. A conceptual Graph Approach to the Generation of Referring Expressions. *Proceedings of IJCAI 2007* : 2456-2461

R. Dale and N. Haddock. 1991. Generating Referring Expressions containing Relations. *Proceedings of Fifth ACL- EACL conference,* 161-166

R. Dale. 1992. *Generating Referring Expressions: Building Descriptions in a Domain of Objects and Processes.* MIT Press

R. Dale and E. Reiter. 1995. Computational Interpretations of the Gricean Maxims in the generation of Referring Expressions. *Cognitive Science* (18): 233 – 263

van Deemter. 2002. Generating Referring Expressions: Boolean Extensions of Incremental Algorithm. *Computational Linguistics* 28(1): 37-52

Coverage-based Evaluation of Parser Generalizability

Tuomo Kakkonen and **Erkki Sutinen**
Department of Computer Science and Statistics
University of Joensuu
P.O. Box 111, FI-80101 Joensuu, Finland
{tuomo.kakkonen,erkki.sutinen}@cs.joensuu.fi

Abstract

We have carried out a series of coverage evaluations of diverse types of parsers using texts from several genres such as newspaper, religious, legal and biomedical texts. We compared the overall coverage of the evaluated parsers and analyzed the differences by text genre. The results indicate that the coverage typically drops several percentage points when parsers are faced with texts on genres other than newspapers.

1 Introduction

The fact that most of the parser evaluation resources employed consist of texts from a single genre constitutes a deficiency in most of the parser evaluations. Evaluations are typically carried out on newspaper texts, *i.e.* on section 23 of the *Penn Treebank* (PTB) (Marcus et al., 1993). A further complication is that many parsing models are trained on the same treebank. Parsers therefore come to be applied to texts from numerous other genres untested. The obvious question that confronts us in these circumstances is: How well will a parser that performs well on financial texts from the Wall Street Journal generalize to other text types?

This present paper addresses parser evaluation from the perspective of coverage. It is a part of a set of evaluations in which selected parsers are evaluated using five criteria: *preciseness*, coverage, *robustness*, *efficiency* and *subtlety*. *Parsing coverage* refers to the ability of a parser to produce an analysis of sentences of naturally occurring free-text. We used parsing coverage to assess the generalizability of the grammars and parsing models and we looked for answers to the following research questions:

- What is the parsing coverage of the evaluated parsers?
- How does the text genre affect the parsing coverage?

Previous work on evaluation methods and resources is discussed in Section 2. Section 3 describes the evaluation method and test settings. In Section 4, we give the results of the experiments. Section 5 concludes with a description of remaining problems and directions for future research.

2 Preliminaries

2.1 Coverage Evaluation

Prasad and Sarkar (2000) observe that the notion of coverage has the following two meanings in the context of parsing. *Grammatical coverage* is the parser's ability to handle different linguistic phenomena, and *parsing coverage* is a measure of the percentage of naturally occurring free text in which a parser can produce a full parse. We divide parsing coverage further into *genre coverage* on different types of texts such as newspapers, religious, biomedicine and fiction.[1]

[1] The classification of texts in terms of domain, genre, register and style is a rather controversial issue (see, for example, discussion by Lee (2001)). A detailed analysis of these issues falls outside of the scope of this paper. We have therefore adopted a simplified approach by indicating differences between texts by using the word genres. One may think of genres (in this sense) as indicating fundamental categorical differences between texts that are revealed in sets of attributes such as domain (e.g. art, science, religion, government), medium

Parsing coverage can be measured as the percentage of input sentences to which a parser is able to assign a parse. No annotated text is needed for performing parsing coverage evaluations. On one hand, it can be argued that coverage alone constitutes a rather weak measure of a parser's performance, and thus of its *generalizability* to diverse text genres. An obvious problem with measuring coverage alone is that a parser returning undetailed and flat analyses will easily get high coverage, whereas a parser that outputs detailed analyses will suffer in covering all the input sentences. Moreover, preciseness and coverage can be seen as conflicting requirements for a parser. Increasing preciseness of the grammar often causes its coverage to decrease; adding more constraints to the grammar causes some of the sentences to be rejected even they are acceptable to users of the language. Loosening the constraints allows more sentences to be parsed, thus increasing the coverage, but at the same time easily leads into overgeneration, problems with disambiguation and decreased preciseness.

On the other hand, the points that we raised above indicate that there is a strong relationship between coverage and preciseness. The aim of syntactic parsers is to analyze whole sentences, not just fragments (constituents/D links) precisely. The connection between coverage and preciseness is clear in the case of sentence level evaluations measures[2]: A sentence that cannot be fully analyzed cannot have a complete match with the correct structure in the evaluation resource. Consequently, we argue that coverage can be used a measure of generalizability; It sets the upper bound for the performance on the sentence-level evaluation measures. However, the evaluation should always be accompanied with data on the preciseness of the parser and the level of detail in its output.

2.2 Previous Coverage and Cross-genre Evaluations

Relatively little work has been done on the empirical evaluation of parsers for text types other than newspaper texts. A key issue in available evaluation materials is the genre homogeneity. Almost all the available resources are based on a single genre (nearly always newspaper texts). This makes it impossible to extrapolate anything useful about the generalizability of the developed grammars and parsing models.

To our knowledge, this experiment is the only one reported in the literature that compares the coverage of a set of parsers for English. The studies that critically examine the genre dependency have come to the same unsurprising conclusion that the text genre has an effect on the parser's performance. The genre dependency of parsers is an accepted fact and has been described by, among others, Sekine (1997) and Gildea (2001). For example, Clegg and Shepherd (2005) have undertaken experiments on biomedical data using the *GENIA treebank*. Laakso (2005) reports experiments on the *CHILDES* corpus of transcribed speech between parents and the children. Mazzei and Lombardo (2004) report cross-training experiments in Italian on newspaper and civil law texts. They observed a dramatic drop of, most commonly, around 10-30 percentage points in the parsing coverage.

2.3 Reasons for the Coverage Drop

Genre dependency is caused by several factors. One is that each text genre is characterized by genre-specific words (Biber, 1993). Another feature of genre dependency is syntactic structure distributions. Baldwin et al. (2004) have conducted one of the rare studies that offer an analysis of the main reasons for the diminished coverage. They experimented with an HPSG grammar that was a created manually based on a corpus of data extracted from informal genres such as conversations about schedules and e-mails about e-commerce. The grammar was used for parsing a random sample of texts from several genres. A diagnosis of failures to parse sentences with full lexical span[3] revealed the following causes for the errors: missing lexical entries (40%), missing constructions (39%), preprocessor errors (4%), fragments (4%), parser failures (4%), and garbage strings (11%). They came to the conclusion that lexical expansion should be the first step in the process of parser enhancement.

(e.g. spoken, written), content (topic, theme) and type (narrative, argumentation, etc.).

[2] For example Yamada & Matsumoto (2003) uses *complete match* metric (the percentage of sentences whose unlabeled D structure is completely correct) to evaluate the sentence-level preciseness of D parsers.

[3] Sentences that contained only words included in the lexicon.

3 Experiments

3.1 Research Approach

In order to investigate the effect of the text genre on the parsing results, we constructed a test corpus of more than 800,000 sentences and divided them into six genres. We parsed these texts by using five parsing systems.

The design of our test settings and materials was guided by our research questions (above). We answered the first question by parsing vast document collections with several state-of-the-art parsing systems and then measuring their parsing coverage on the data. Because we had divided our purpose-built test set into genre-specific subsets, this allowed us to measure the effects of genre variance and so provide an answer to the second research question. We also included two parsers that had been developed in the 1990s to evaluate the extent to which progress has been made in parsing technology in genre dependency and parsing coverage.

3.2 Evaluation Metric and Measures

The most important decision in parsing coverage evaluation is how the distinction between a covered and uncovered sentence is made. This has to be defined separately for each parser and the definition depends on the type of output. We implemented a set of Java tools to record the statistics from the parsers' outputs. In addition to completely failed parses, we recorded information about incomplete analyses and the number of times the parsers crashed or terminated during parsing.

3.3 Materials

The test set consisted of 826,485 sentences divided into six sub-corpora. In order to cover several genres and to guarantee the diversity of the text types, we sourced a diversity of materials from several collections. There are six sub-corpora in the material and each covers one of the following genres: newspaper, legislation, fiction, non-fiction, religion and biomedicine.

Table 1 shows the sub-corpora and the figures associated with each corpus. In total there were 15,385,855 tokens. The style of the newspaper texts led us to make an initial hypothesis that a similar performance would probably be achievable with non-fiction texts, and we suspected that the legislative and fiction texts might be more difficult to parse because of the stylistic idiosyncrasies involved. Biomedical texts also contained a considerable number of words that are probably not found in the lexicons. These two difficulties were compounded in the religious texts, and the average length of the religion sub-corpus was far higher than the average.

Table 1. The test sets.

Genre	Description	No. of sentences	Avg. length
Legislation	Discussions of the Canadian Parliament	390,042	17.2
Newspaper	Texts from several newspapers	217,262	19.5
Fiction	Novels from the 20th and 21st century	97,156	15.9
Non-fiction	Non-fiction books from the 20th and 21st century	61,911	21.9
Religion	The Bible, the Koran, the Book of Mormon	45,459	27.1
Biomedicine	Abstracts from biomedical journals	14,655	21.6
TOTAL		826,485	18.6

3.4 The Parsers

We included both dependency (D)- and phrase structure (PS)-based systems in the experiment. The parsers use a *Probabilistic Context-free* Grammar (PCFG), *Combinatory Categorial Grammar* (CCG), a semi-context sensitive grammar and a D-based grammar.

Apple Pie Parser (APP) (v. 5.9, 4 April 1997) is a bottom-up probabilistic chart parser which finds the analysis with the best score by means of best-first search algorithm (Sekine, 1998). It uses a semi-context sensitive grammar obtained automatically from the PTB. The parser outputs a PS analysis consisting of 20 syntactic tags. No word-level analysis is assigned. We regard a sentence as having been covered if APP finds a single S non-terminal which dominates the whole sentence and if it does not contain any X tags which would indicate constituents of unrecognized category.

C&C Parser (v. 0.96, 23 November 2006) is based on a CCG. It applies log-linear probabilistic tagging and parsing models (Clark and Curran, 2004). Because the parser marks every output as

either parsed or failed, evaluation of failed parses is straightforward. Fragmented parses were detected from the *grammatical relations* (GR) output. Because GR representations can form cycles, an analysis was not required to have a unique root. Instead, a parse was regarded as being incomplete if, after projecting each GR to a graph allowing cycles, more than one connected set (indicating a fragmented analysis) was found.

MINIPAR (unknown version, 1998) is a principle-based parser applying a distributed chart algorithm and a D-style grammar (Lin, 1998). The syntactic tagset comprises 27 grammatical relation types and word and phrase types are marked with 20 tags. A sentence is regarded as having been covered by MINIPAR if a single root is found for it that is connected to all the words in the sentence through a path. The root should in addition be assigned with a phrase/sentence type marker.

Stanford Parser (referred in the remainder of this text as SP) (v. 1.5.1, 30 May 2006) can use both an unlexicalized and lexicalized PCFGs (Klein and Manning, 2003). This parser uses a CYK search algorithm and can output both D and PS analyses (de Marneffe et al., 2006). We ran the experiment on the unlexicalized grammar and carried out the evaluation on the D output consisting of 48 D types. We regard a sentence as having been covered by SP in a way similar to that in MINIPAR: the sentence is covered if the D tree returned by the parser has a single root node in which there is a path to all the other nodes in the tree.

StatCCG (Preliminary public release, 14 January 2004) is a statistical parser for CCG that was developed by Julia Hockenmaier (2003). In contrast to C&C, this parser is based on a generative probabilistic model. The lexical category set has around 1,200 types, and there are four atomic types in the syntactic description. StatCCG marks every relevant sentence as 'failed' or 'too long' in its output. We were therefore able to calculate the failed parses directly from the system output. We regarded parses as being partially covered when no sentence level non-terminal was found.

3.5 Test Settings

We wanted to create similar and equal conditions for all parsers throughout the evaluation. Moreover, language processing applications that involve parsing must incorporate practical limits on resource consumption.[4] Hence, we limited the use of memory to the same value for all the parsers and experiments.[5] We selected 650 MB as the upper limit. It is a realistic setting for free working memory in a typical personal computer with 1 GB memory.

4 Results

Table 2 summarizes the results of the experiments. The parsing coverage of the parsers for each of the sub-corpora is reported separately. Total figures are given for both parser and sub-corpus level. In Table 3, the coverage figures are further broken down to indicate the percentage of the analyses that failed or were incomplete or those occasions on which the parser crashed or terminated during the process.

The five parsers were able to cover, on average, 88.8% of the sentences. The coverage was, unsurprisingly, highest on the newspaper genre. The lowest average coverage was achieved on the religion genre. The difficulties in parsing the religious texts are attributable at least in part to the length of the sentences in the sub-corpus (on average 27.1 words per sentence), which was the highest over all the genres. Contrary to our expectation, the biomedical genre, with its specialist terminology, was not the most difficult genre for the parsers.

If one excludes the one-word sentences from the legislation dataset, SP had the best coverage and best generalizability over the text genres. APP was the second best performer in this experiment, both in coverage and generalizability. While APP produces shallow parses, this helps it to obtain a high coverage. Moreover, comparing the F-scores reported in the literature for the five parsers revealed that the F-score (70.1) of this parser was more than 10 percentage points lower than the score of the second worst parser MINIPAR. Thus, it is obvious that the high coverage in APP is achieved at the cost of preciseness and lack of detail in the output.

[4] In addition, parsing in the order of hundreds of thousands of sentences with five parsers takes thousands of hours of processor time. It was therefore necessary for us to limit the memory consumption in order to be able to run the experiments in parallel.

[5] Several methods were used for limiting the memory usage depending on the parser. For example, in the Java-based parsers, the limit was set on the size of the Java heap.

Table 2. Comparison of the parsing results for each sub-corpus and parser. "Average" column gives the average of the coverage figures for the six genres weighted according to the number of sentences in each genre. The column labeled "Generalizability" shows the drop of the coverage in the lowest-scoring genre compared to the coverage in the newspaper genre.

Parser	*Newspaper*	*Legislation*	*Fiction*	*Non-fiction*	*Religion*	*Biomedicine*	*Average*	*Generalizability*
APP	99.8	98.9	97.5	96.4	93.1	98.9	98.5	6.7
C&C	87.8	84.9	86.0	81.2	75.5	84.8	85.0	14.0
MINIPAR	88.0	68.8	68.0	71.5	34.4	70.1	72.1	60.9
SP*	99.8	99.5	98.0	98.3	98.9	98.5	99.2	1.8
StatCCG	96.7	85.2	87.7	86.7	94.0	83.3	89.1	13.9
Average	94.4	87.5	87.4	86.8	79.2	87.1	88.8	19.5

*SP experienced a coverage drop of tens of percentage points in comparison to other genres on the Hansard dataset. This was caused mainly by a single issue: the dataset contained a number of sentences that contained only a single word – sentences such as "Nay.", "Agreed.", "No." and so on. Because no root node is assigned to D analysis by SP, the parser did not return any analysis for such sentences. These sentences were omitted from the evaluation. When the sentences were included, the coverage on legislation data was 59.5% and the average was 73.4%.

Table 3**.** Breakdown of the failures. All the results are reported as a percentage of the total number of sentences. Column 'Incomplete' reports the proportion of sentences that were parsed, but the analysis was not full. Column 'Failed' shows those cases in which the parser was not able to return a parse. Column 'Terminated' shows the proportion of the cases in which the parser crashed or terminated during the process of parsing a sentence.

Parser	*Incomplete*	*Failed*	*Terminated*
APP	1.5	0.0	0.001
C&C	12.8	2.2	0.006
MINIPAR	27.9	0.0	0.009
SP	0.5	0.4	0.002
StatCCG	9.6	1.4	0.000
Average	10.5	0.8	0.004

While StatCCG outperformed C&C parser by 4.1 percentage points in average coverage, the two CCG-based parsers achieved a similar generalizability. StatCCG was the most stable parser in the experiment. It did not crash or terminate once on the test data.

The only parser based on a manually-constructed grammar, MINIPAR, had the lowest coverage and generalizability. MINIPAR also proved to have stability problems. While this parser achieved an 88.0% coverage with the newspaper corpus, its performance dropped over 10 percentage points with other corpora. Its coverage was only 34.4% with the religion genre. The most commonly occurring type of problem with this data was a fragmented analysis occasioned by sentences beginning with an 'And' or 'Or' that was not connected to any other words in the parse tree.

5 Conclusion

This paper describes our experiments in parsing diverse text types with five parsers operating with four different grammar formalisms. To our knowledge, this experiment is the only large-scale comparison of the coverage of a set of parsers for English reported in the literature. On average, the parsing coverage of the five parsers on newspaper texts was 94.4%. The average dropped from 5.6 to 15.2 percentage points on the other five text genres. The lowest average scores were achieved on the religion test set.

In comparison to MINIPAR, the results indicate that the coverage of the newer parsers has improved. The good performance of the APP may partly be explained by a rather poor preciseness: the rate of just over 70% is much lower than that of other parsers. APP also produces a shallow analysis that enables it to achieve a high coverage.

One observation that should be made relates to the user friendliness and documentation of the parsing systems. The parsing of a vast collection of texts using several parsing systems was neither simple nor straightforward. To begin with, most of the parsers crashed at least once during the course of the experiments. The C&C parser, for example, terminates when it encounters a sentence with two spaces between words. It would be far more con-

venient for users if such sentences were automatically skipped or normalized.

While another feature is that all the parsers have a set of parameters that can be adjusted, the accompanying documentation about their effects is in many cases insufficiently detailed. From the NLP practitioner's point of view, the process of selecting an appropriate parser for a given task is complicated by the fact that the output format of a parser is frequently described in insufficient detail. It would also be useful in many NLP applications if the parser were able to indicate whether or not it could parse a sentence completely. It would also be optimal if a confidence score indicating the reliability of the returned analysis could be provided.

The most obvious directions for work of this kind would include other text genres, larger collections of texts and more parsers. One could also pinpoint the most problematic types of sentence structures by applying error-mining techniques to the results of the experiments.

References

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the 4th Language Resources and Evaluation Conference (LREC)*, Lisbon, Portugal.

Douglas Biber. 1993. Using Register-diversified Corpora for General Language Studies. *Computational Linguistics*, 19(2):219–241.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd ACL*, Barcelona, Spain.

Andrew B. Clegg and Adrian J. Shepherd. 2005. Evaluating and Integrating Treebank Parsers on a Biomedical Corpus. In *Proceedings of the Workshop on Software at the 43rd ACL*, Ann Arbor, Michigan, USA.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the 5th LREC*, Genoa, Italy.

Daniel Gildea. 2001. Corpus Variation and Parser Performance. In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing, Pittsburgh, Pennsylvania, USA.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD Dissertation, University of Edinburgh, UK.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st ACL*, Sapporo, Japan.

Aarre Laakso. On Parsing CHILDES. 2005. In Proceedings of the Second Midwest Computational Linguistics Colloquium. Columbus, Ohio, USA,

David Lee. 2001. Genres, Registers, Text Types, Domains, and Styles: Clarifying the Concepts and Navigating a Path through the BNC Jungle. *Language Learning & Technology*, 5(3):37–72.

Dekang Lin. 1998. Dependency-Based Evaluation of MINIPAR. In *Proceedings of the 1st LREC*, Granada, Spain.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Alessandro Mazzei and Vincenzo Lombardo. 2004. A Comparative Analysis of Extracted Grammars. In *Proceedings of the 16th European Conference on Artificial Intelligence*, Valencia, Spain.

Rashmi Prasad and Anoop Sarkar. 2000. Comparing Test-suite Based Evaluation and Corpus-based Evaluation of a Wide Coverage Grammar for English. In *Proceedings of the Using Evaluation within HLT Programs: Results and Trends Workshop at the 2nd LREC*, Athens, Greece.

Geoffrey Sampson, editor. 1995. *English for the Computer: The Susanne Corpus and Analytic Scheme*. Oxford University Press, Oxford, UK.

Satoshi Sekine. 1997. The Domain Dependence of Parsing. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington, DC, USA.

Satoshi Sekine. 1998. *Corpus-based Parsing and Sublanguage Studies*. PhD Thesis. New York University, New York, USA.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, Nancy, France.

Learning Reliability of Parses for Domain Adaptation of Dependency Parsing

Daisuke Kawahara and **Kiyotaka Uchimoto**
National Institute of Information and Communications Technology,
3-5 Hikaridai Seika-cho Soraku-gun, Kyoto, 619-0289, Japan
{dk, uchimoto}@nict.go.jp

Abstract

The accuracy of parsing has exceeded 90% recently, but this is not high enough to use parsing results practically in natural language processing (NLP) applications such as paraphrase acquisition and relation extraction. We present a method for detecting reliable parses out of the outputs of a single dependency parser. This technique is also applied to domain adaptation of dependency parsing. Our goal was to improve the performance of a state-of-the-art dependency parser on the data set of the domain adaptation track of the CoNLL 2007 shared task, a formidable challenge.

1 Introduction

Dependency parsing has been utilized in a variety of natural language processing (NLP) applications, such as paraphrase acquisition, relation extraction and machine translation. For newspaper articles, the accuracy of dependency parsers exceeds 90% (for English), but it is still not sufficient for practical use in these NLP applications. Moreover, the accuracy declines significantly for out-of-domain text, such as weblogs and web pages, which have commonly been used as corpora. From this point of view, it is important to consider the following points to use a parser practically in applications:

- to select reliable parses, especially for knowledge acquisition,
- and to adapt the parser to new domains.

This paper proposes a method for selecting reliable parses from parses output by a single dependency parser. We do not use an ensemble method based on multiple parsers, but use only a single parser, because speed and efficiency are important when processing a massive volume of text. The resulting highly reliable parses would be useful to automatically construct dictionaries and knowledge bases, such as case frames (Kawahara and Kurohashi, 2006). Furthermore, we incorporate the reliable parses we obtained into the dependency parser to achieve domain adaptation.

The CoNLL 2007 shared task tackled domain adaptation of dependency parsers for the first time (Nivre et al., 2007). Sagae and Tsujii applied an ensemble method to the domain adaptation track and achieved the highest score (Sagae and Tsujii, 2007). They first parsed in-domain unlabeled sentences using two parsers trained on out-of-domain labeled data. Then, they extracted identical parses that were produced by the two parsers and added them to the original (out-of-domain) training set to train a domain-adapted model.

Dredze et al. yielded the second highest score[1] in the domain adaptation track (Dredze et al., 2007). However, their results were obtained without adaptation. They concluded that it is very difficult to substantially improve the target domain performance over that of a state-of-the-art parser. To confirm this, we parsed the test set (CHEM) of the domain adaptation track by using one of the best dependency parsers, second-order MSTParser (McDonald et al.,

[1] Dredze et al. achieved the second highest score on the CHEM test set for unlabeled dependency accuracy.

2006)[2]. Though this parser was trained on the provided out-of-domain (Penn Treebank) labeled data, surprisingly, its accuracy slightly outperformed the highest score achieved by Sagae and Tsujii (unlabeled dependency accuracy: $83.58 > 83.42$ (Sagae and Tsujii, 2007)). Our goal is to improve a state-of-the-art parser on this domain adaptation track.

Dredze et al. also indicated that unlabeled dependency parsing is not robust to domain adaptation (Dredze et al., 2007). This paper therefore focuses on unlabeled dependency parsing.

2 Related Work

We have already described the domain adaptation track of the CoNLL 2007 shared task. For the multilingual dependency parsing track, which was the other track of the shared task, Nilsson et al. achieved the best performance using an ensemble method (Hall et al., 2007). They used a method of combining several parsers' outputs in the framework of MST parsing (Sagae and Lavie, 2006). This method does not select parses, but considers all the output parses with weights to decide a final parse of a given sentence.

Reichart and Rappoport also proposed an ensemble method to select high-quality parses from the outputs of constituency parsers (Reichart and Rappoport, 2007a). They regarded parses as being of high quality if 20 different parsers agreed. They did not apply their method to domain adaptation or other applications.

Reranking methods for parsing have a relation to parse selection. They rerank the n-best parses that are output by a generative parser using a lot of lexical and syntactic features (Collins and Koo, 2005; Charniak and Johnson, 2005). There are several related methods for 1-best outputs, such as revision learning (Nakagawa et al., 2002) and transformation-based learning (Brill, 1995) for part-of-speech tagging. Attardi and Ciaramita proposed a method of tree revision learning for dependency parsing (Attardi and Ciaramita, 2007).

As for the use of unlabeled data, self-training methods have been successful in recent years. McClosky et al. improved a state-of-the-art constituency parser by 1.1% using self-training (McClosky et al., 2006a). They also applied self-training to domain adaptation of a constituency parser (McClosky et al., 2006b). Their method simply adds parsed unlabeled data without selecting it to the training set. Reichart and Rappoport applied self-training to domain adaptation using a small set of in-domain training data (Reichart and Rappoport, 2007b).

[2]http://sourceforge.net/projects/mstparser/

Table 1: Labeled and unlabeled data provided for the shared task. The labeled PTB data is used for training, and the labeled BIO data is used for development. The labeled CHEM data is used for the final test.

name	source	labeled	unlabeled
PTB	Penn Treebank	18,577	1,625,606
BIO	Penn BioIE	200	369,439
CHEM	Penn BioIE	200	396,128

Van Noord extracted bilexical preferences from a Dutch parsed corpus of 500M words without selection (van Noord, 2007). He added some features into an HPSG (head-driven phrase structure grammar) parser to consider the bilexical preferences, and obtained an improvement of 0.5% against a baseline.

Kawahara and Kurohashi extracted reliable dependencies from automatic parses of Japanese sentences on the web to construct large-scale case frames (Kawahara and Kurohashi, 2006). Then they incorporated the constructed case frames into a probabilistic dependency parser, and outperformed their baseline parser by 0.7%.

3 The Data Set

This paper uses the data set that was used in the CoNLL 2007 shared task (Nivre et al., 2007). Table 1 lists the data set provided for the domain adaptation track.

We pre-processed all the unlabeled sentences using a conditional random fields (CRFs)-based part-of-speech tagger. This tagger is trained on the PTB training set that consists of 18,577 sentences. The features are the same as those in (Ratnaparkhi, 1996). As an implementation of CRFs, we used CRF++[3]. If a method of domain adaptation is applied to the tagger, the accuracy of parsing unlabeled sentences will improve (Yoshida et al., 2007). This

[3]http://crfpp.sourceforge.net/

paper, however, does not deal with domain adaptation of a tagger but focuses on that of a parser.

4 Learning Reliability of Parses

Our approach assesses automatic parses of a single parser in order to select only reliable parses from them. We compare automatic parses and their gold-standard ones, and regard accurate parses as positive examples and the remainder as negative examples. Based on these examples, we build a binary classifier that classifies each sentence as reliable or not. To precisely detect reliable parses, we make use of several linguistic features inspired by the notion of controlled language (Mitamura et al., 1991). That is to say, the reliability of parses is judged based on the degree of sentence difficulty.

Before describing our base dependency parser and the algorithm for detecting reliable parses, we first explain the data sets used for them. We prepared the following three labeled data sets to train the base dependency parser and the reliability detector.

PTB_base_train: training set for the base parser: 14,862 sentences
PTB_rel_train: training set for reliability detector: 2,500 sentences[4]
BIO_rel_dev: development set for reliability detector: 200 sentences (= labeled BIO data)

PTB_base_train is used to train the base dependency parser, and PTB_rel_train is used to train our reliability detector. BIO_rel_dev is used for tuning the parameters of the reliability detector.

4.1 Base Dependency Parser

We used the MSTParser (McDonald et al., 2006), which achieved top results in the CoNLL 2006 (CoNLL-X) shared task, as a base dependency parser. To enable second-order features, the parameter *order* was set to 2. The other parameters were set to default. We used PTB_base_train (14,862 sentences) to train this parser.

4.2 Algorithm to Detect Reliable Parses

We built a binary classifier for detecting reliable sentences from a set of automatic parses produced by the base dependency parser.

We used support vector machines (SVMs) as a binary classifier with a third-degree polynomial kernel. We parsed PTB_rel_train (2,500 sentences) using the base parser, and evaluated each sentence with the metric of unlabeled dependency accuracy. We regarded the sentences whose accuracy is better than a threshold, τ, as positive examples, and the others as negative ones. In this experiment, we set the accuracy threshold τ at 100%. As a result, 736 out of 2,500 examples (sentences) were judged to be positive.

To evaluate the reliability of parses, we take advantage of the following features that can be related to the difficulty of sentences.

sentence length: The longer the sentence is, the poorer the parser performs (McDonald and Nivre, 2007). We determine sentence length by the number of words.

dependency lengths: Long-distance dependencies exhibit bad performance (McDonald and Nivre, 2007). We calculate the average of the dependency length of each word.

difficulty of vocabulary: It is hard for supervised parsers to learn dependencies that include low-frequency words. We count word frequencies in the training data and make a word list in descending order of frequency. For a given sentence, we calculate the average frequency rank of each word.

number of unknown words: Similarly, dependency accuracy for unknown words is notoriously poor. We count the number of unknown words in a given sentence.

number of commas: Sentences with multiple commas are difficult to parse. We count the number of commas in a given sentence.

number of conjunctions (*and/or*): Sentences with coordinate structures are also difficult to parse (Kurohashi and Nagao, 1994). We count the number of coordinate conjunctions (*and/or*) in a given sentence.

To apply these features to SVMs in practice, the numbers are binned at a certain interval for each feature. For instance, the number of conjunctions is split into four bins: 0, 1, 2 and more than 2.

[4] 1,215 labeled PTB sentences are left as another development set for the reliability detector, but they are not used in this paper.

Table 2: Example BIO sentences judged as reliable. The underlined words have incorrect modifying heads.

dep. accuracy	sentences judged as reliable
12/12 (100%)	No mutations resulting in truncation of the APC protein were found .
12/13 (92%)	Conventional imaging techniques did not show two in 10 of these patients .
6/6 (100%)	Pancreatic juice was sampled endoscopically .
11/12 (92%)	The specificity of p53 mutation for pancreatic cancer is very high .
9/10 (90%)	K-ras mutations are early genetic changes in colon cancer .

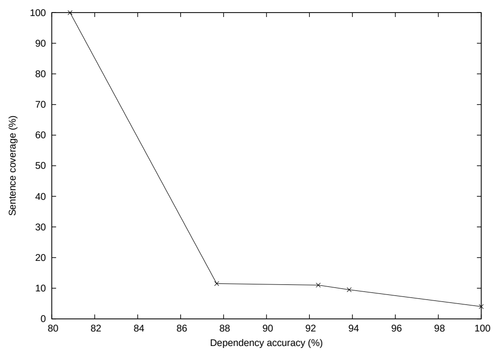


Figure 1: Accuracy-coverage curve on BIO_rel_dev.

4.3 Experiments on Detecting Reliable Parses

We conducted an experiment on detecting the reliability of parses. Our detector was applied to the automatic parses of BIO_rel_dev, and only reliable parses were selected from them. When parsing this set, the POS tags contained in the set were substituted with automatic POS tags because it is preferable to have the same environment as when applying the parser to unlabeled data.

We evaluated unlabeled dependency accuracy of the extracted parses. The accuracy-coverage curve shown in Figure 1 was obtained by changing the soft margin parameter C [5] of SVMs from 0.0001 to 10. In this figure, the coverage is the ratio of selected sentences out of all the sentences (200 sentences), and the accuracy is unlabeled dependency accuracy. A coverage of 100% indicates that the accuracy of 200 sentences without any selection was 80.85%.

If the soft margin parameter C is set to 0.001, we can obtain 19 sentences out of 200 at a dependency accuracy of 93.85% (183/195). The average sentence length was 10.3 words. Out of obtained 19 sentences, 14 sentences achieved a dependency accuracy of 100%, and thus the precision of the reliability detector itself was 73.7% (14/19). Out of 200 sentences, 36 sentences were correctly parsed by the base parser, and thus the recall is 38.9% (14/36).

Table 2 shows some sentences that were evaluated as reliable using the above setting (C = 0.001). Major errors were caused by prepositional phrase (PP)-attachment. To improve the accuracy of detecting reliable parses, it would be necessary to consider the number of PP-attachment ambiguities in a given sentence as a feature.

5 Domain Adaptation of Dependency Parsing

For domain adaptation, we adopt a self-training method. We combine in-domain unlabeled (automatically labeled) data with out-of-domain labeled data to make a training set. There are many possible methods for combining unlabeled and labeled data (Daumé III, 2007), but we simply concatenate unlabeled data with labeled data to see the effectiveness of the selected reliable parses. The in-domain unlabeled data to be added are selected by the reliability detector. We set the soft margin parameter at 0.001 to extract highly reliable parses. As mentioned in the previous section, the accuracy of selected parses was approximately 94%.

We parsed the unlabeled sentences of BIO and CHEM (approximately 400K sentences for each) using the base dependency parser that is trained on the entire PTB labeled data. Then, we applied the reliability detector to these parsed sentences to obtain 31,266 sentences for BIO and 31,470 sentences for CHEM. We call the two sets of obtained sentences "BIO pool" and "CHEM pool".

For each training set of the experiments described below, a certain number of sentences are randomly selected from the pool and combined with the entire out-of-domain (PTB) labeled data.

5.1 Experiment on BIO Development Data

We first conducted an experiment of domain adaptation using the BIO development set.

[5] A higher soft margin value allows more classification errors, and thus leads to the increase of recall and the decrease of precision.

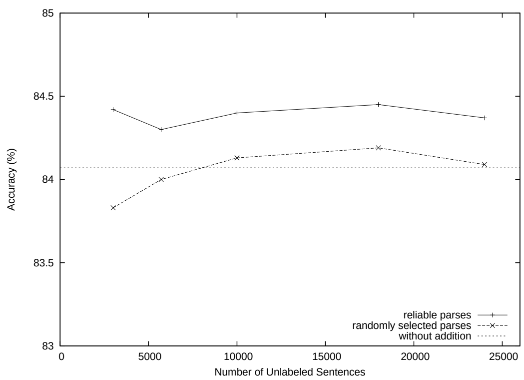


Figure 2: Dependency accuracies on BIO when the number of added unlabeled data is changed.

Figure 2 shows how the accuracy changes when the number of added reliable parses is changed. The solid line represents our proposed method, and the dotted line with points represents a baseline method. This baseline is a self-training method that simply adds unlabeled data without selection to the PTB labeled data. Each experimental result is the average of five trials done to randomly select a certain number of parses from the BIO pool. The horizontal dotted line (84.07%) represents the accuracy of the parser without adding unlabeled data (trained only on the PTB labeled data).

From this figure, we can see that the proposed method always outperforms the baseline by approximately 0.4%. The best accuracy was achieved when 18,000 unlabeled parses were added. However, if more than 18,000 sentences are added, the accuracy declines. This can be attributed to the balance of the number of labeled data and unlabeled data. Since the number of added unlabeled data is more than the number of labeled data, the entire training set might be unreliable, though the accuracy of added unlabeled data is relatively high. To address this problem, it is necessary to weigh labeled data or to change the way information from acquired unlabeled data is handled.

5.2 Experiment on CHEM Test Data

The addition of 18,000 sentences showed the highest accuracy for the BIO development data. To adapt the parser to the CHEM test set, we used 18,000 reliable unlabeled sentences from the CHEM pool with the PTB labeled sentences to train the parser. Table 3 lists the experimental results. In this table, the third row lists the three highest scores of the domain adaptation track of the CoNLL 2007 shared task.

Table 3: Experimental results on CHEM test data.

system	accuracy
PTB+unlabel (18,000 sents.)	84.12
only PTB (baseline)	83.58
1st (Sagae and Tsujii, 2007)	83.42
2nd (Dredze et al., 2007)	83.38
3rd (Attardi et al., 2007)	83.08

The baseline parser was trained only on the PTB labeled data (as described in Section 1). The proposed method (PTB+unlabel (18,000 sents.)) outperformed the baseline by approximately 0.5%, and also beat all the systems submitted to the domain adaptation track. These systems include an ensemble method (Sagae and Tsujii, 2007) and an approach of tree revision learning with a selection method of only using short training sentences (shorter than 30 words) (Attardi et al., 2007).

6 Discussion and Conclusion

This paper described a method for detecting reliable parses out of the outputs of a single dependency parser. This technique was also applied to domain adaptation of dependency parsing.

To extract reliable parses, we did not adopt an ensemble method, but used a single-parser approach because speed and efficiency are important in processing a gigantic volume of text to benefit knowledge acquisition. In this paper, we employed the MSTParser, which can process 3.9 sentences/s on a XEON 3.0GHz machine in spite of the time complexity of $O(n^3)$. If greater efficiency is required, it is possible to apply a pre-filter that removes long sentences (e.g., longer than 30 words), which are seldom selected by the reliability detector. In addition, our method does not depend on a particular parser, and can be applied to other state-of-the-art parsers, such as Malt Parser (Nivre et al., 2006), which is a feature-rich linear-time parser.

In general, it is very difficult to improve the accuracy of the best performing systems by using unlabeled data. There are only a few successful studies, such as (Ando and Zhang, 2005) for chunking and (McClosky et al., 2006a; McClosky et al., 2006b) on constituency parsing. We succeeded in boosting the accuracy of the second-order MST parser, which is

a state-of-the-art dependency parser, in the CoNLL 2007 domain adaptation task. This was a difficult challenge as many participants in the task failed to obtain any meaningful gains from unlabeled data (Dredze et al., 2007). The key factor in our success was the extraction of only reliable information from unlabeled data.

However, that improvement was not satisfactory. In order to achieve more gains, it is necessary to exploit a much larger number of unlabeled data. In this paper, we adopted a simple method to combine unlabeled data with labeled data. To use this method more effectively, we need to balance the labeled and unlabeled data very carefully. However, this method is not scalable because the training time increases significantly as the size of a training set expands. We can consider the information from more unlabeled data as features of machine learning techniques. Another approach is to formalize a probabilistic model based on unlabeled data.

References

Rie Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of ACL2005*, pages 1–9.

Giuseppe Attardi and Massimiliano Ciaramita. 2007. Tree revision learning for dependency parsing. In *Proceedings of NAACL-HLT2007*, pages 388–395.

Giuseppe Attardi, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of EMNLP-CoNLL2007*, pages 1112–1118.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing. *Computational Linguistics*, 21(4):543–565.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL2005*, pages 173–180.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL2007*, pages 256–263.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João V. Graça, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of EMNLP-CoNLL2007*, pages 1051–1055.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryigit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of EMNLP-CoNLL2007*, pages 933–939.

Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of HLT-NAACL2006*, pages 176–183.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of HLT-NAACL2006*, pages 152–159.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of COLING-ACL2006*, pages 337–344.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL2007*, pages 122–131.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL-X*, pages 216–220.

Teruko Mitamura, Eric Nyberg, and Jaime Carbonell. 1991. An efficient interlingua translation system for multi-lingual document production. In *Proceedings of MT Summit III*, pages 55–61.

Tetsuji Nakagawa, Taku Kudo, and Yuji Matsumoto. 2002. Revision learning and its application to part-of-speech tagging. In *Proceedings of ACL2002*, pages 497–504.

Joakim Nivre, Johan Hall, Jens Nilsson, Gül sen Eryi git, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL-X*, pages 221–225.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL2007*, pages 915–932.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP1996*, pages 133–142.

Roi Reichart and Ari Rappoport. 2007a. An ensemble method for selection of high quality parses. In *Proceedings of ACL2007*, pages 408–415.

Roi Reichart and Ari Rappoport. 2007b. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *Proceedings of ACL2007*, pages 616–623.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Companion Volume to HLT-NAACL2006*, pages 129–132.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of EMNLP-CoNLL2007*, pages 1044–1050.

Gertjan van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of IWPT2007*, pages 1–10.

Kazuhiro Yoshida, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Ambiguous part-of-speech tagging for improving accuracy and domain portability of syntactic parsers. In *Proceedings of IJCAI-07*, pages 1783–1788.

Resolving Ambiguities of Chinese Conjunctive Structures by Divide-and-conquer Approaches

Duen-Chi Yang, Yu-Ming Hsieh, Keh-Jiann Chen
Institute of Information Science, Academia Sinica, Taipei
{ydc, morris, kchen}@iis.sinica.edu.tw

Abstract

This paper presents a method to enhance a Chinese parser in parsing conjunctive structures. Long conjunctive structures cause long-distance dependencies and tremendous syntactic ambiguities. Pure syntactic approaches hardly can determine boundaries of conjunctive phrases properly. In this paper, we propose a divide-and-conquer approach which overcomes the difficulty of data-sparseness of the training data and uses both syntactic symmetry and semantic reasonableness to evaluate ambiguous conjunctive structures. In comparing with the performances of the PCFG parser without using the divide-and-conquer approach, the precision of the conjunctive boundary detection is improved from 53.47% to 83.17%, and the bracketing f-score of sentences with conjunctive structures is raised up about 11 %.

1 Introduction

Parsing a sentence with long conjunctive structure is difficult, since it is inadequate for a context-free grammar to represent context-sensitive-like coordination structures, such as "a b c… and a' b' c'… ". It causes long-distance dependencies and tremendous syntactic ambiguities (a large number of alternatives). Pure syntactic approaches cannot determine boundaries of conjunctive phrases properly. It is obvious that both syntactic and semantic information are necessary for resolving ambiguous boundaries of conjunctive structures.

Some analysis methods of the detection of conjunctive structures have been studied for a while. Despite of using different resources and tools, these methods mainly make use of the similarity of words or word categories on both sides of conjunctive structure (Agarwal et al., 1992; Kurohashi et al., 1994; Delden, 2002; Steiner 2003). They assumed that two sides of conjuncts should have similar syntactic and semantic structures. Some papers also suggest that certain key word patterns can be used to decide the boundaries (Wu 2003). Agarwal et al. (1992) used a semantic tagger and a syntactic chunker to label syntactic and semantic chunks. And then they defined multi-level (category to category or semantic type to semantic type) similarity matching to find the structure boundaries. Delden (2002) included semantic analysis by applying WordNet (Miller 1993) information. These presented methods used similarity measures heuristically according to the property of the languages. However detecting conjunctive boundaries with a similar method in Chinese may meet some problems, since a Chinese word may play different syntactic functions without inflection. It results that syntactic symmetry is not enough to resolve ambiguities of conjunctive structures and semantic reasonableness is hard to be evaluated. Therefore we propose a divide-and-conquer approach which takes the advantage of using structure information of partial sentences located at both sides of conjunction. Furthermore we believe that simple cases can be solved by simple methods which are efficient and only complex cases require deep syntactic and semantic analysis. Therefore we develop an algorithm to discriminate simple cases and complex cases first. We then use a sophisticated algorithm to handle complex cases only.

For simple cases, we use conventional pattern matching approach to speedup process. For complex conjunctive structures, we propose a divide-and-conquer approach to resolve the problem. An input sentence with complex conjunctive structure

is first divided into two parts, one to the left of the conjunctive and one to the right, and then parsed independently to detect possible candidates of two conjuncts. The particular property of complex conjunctive structures of Chinese language allows us to parse and to produce syntactic structures of two partial sentences, since according to our observations and experiments the syntactic structures of partial sentences at either side of a complex conjunctive construction are grammatical most of the times. Figure 1 shows an instance. The parsing results not only reduce the possible ambiguous boundaries but also provide global structural information for checking the properness of both sides of conjunctive structure. Another important point worth mentioning is that since the size of available Treebank is small, a two-stage approach is proposed to resolve the data sparseness problems in evaluating syntactic symmetry and semantic reasonableness. At the first stage, a Conditional Random Fields model is trained and used to generate a set of candidate boundaries. At the second stage, a word-association model is trained from a gigaword corpus to evaluate the semantic properness of candidates. The proposed divide-and-conquer algorithm avoids parsing full complex conjunctive structures and handles conjunctive structures with deep structural and semantic analysis.

The extraction method for context-dependent rules is described in Section 2 and detail of the divide-and-conquer approach is stated in Section 3. In Section 4, we introduce our experimental environment and show the results of our experiment. We also make some discussions about our observations in Section 4. Finally, we offer our conclusion and future work in Section 5.

2 Boundary Detection for Simple Conjunctive Phrases

The aim of this phase of approach is to determine if simple conjunctive phrases exist in input sentences and then identify their boundaries by matching context-dependent rules. To derive a set of context-dependent rules for conjunctive phrases, a naïve approach is to extract all conjunctive patterns with their contextual constraints from Treebank. However such a set of extracted rules suffers a low coverage rate, since limited size of training data causes zero frequency of long n-gram PoS patterns.

2.1 Rule extraction and generalization

Agarwal et al., (1992), Kurohashi et al., (1994), and Delden (2002) had shown that the properties of likeness and symmetry in both syntactic types and lengths for example, exist in most conjunctive cases. Hence we use both properties as the conditions in deciding boundaries of conjunctive phrases. When we observe Sinica Treebank (Chen et al., 2003), we also find that this property is more obvious in simple conjunctive cases than in complex cases.

First, we use a simple algorithm to detect the boundaries of completely symmetric conjunctive phrases. If PoS patterns of “A B C and A B C” or “A B and A B” occurred in the input sentence, we consider patterns of such structures are legitimate conjunctive structures regardless whether the PoS sequences “A B C and A B C” or “A B and A B” ever occurred in the Treebank. For other cases we use context-dependent rule patterns to determine boundaries of conjunctive structures.

Statistical context-dependent PoS-based rule patterns are extracted automatically from Sinica Treebank. Each rule contains the PoS pattern of a conjunctive phrase and its left/right contextual constraints. The occurrence frequency of the rule and its correct identification rate are also associated. e.g. [VC] (Na Caa Nc) [DE][1] ; 12; 11

This rule says that PoS sequence Na Caa Nc forms a conjunctive phrase when its left context is a VC and its right context is a DE. Such pattern occurred 12 times in the training corpus and 11 out of 12 times (Na Caa Nc) are correct conjunctive phrases.

Context-dependent rule patterns are generated and generalized by the following procedure.

Rule Generation and Generalization

For each conjunctive structure in the Treebank, we consider a window pattern of at most 9 words. This pattern contains conjunction in the center and at most 4 words at each side of the conjunction. The PoS sequence of these 9 words forms a context-dependent rule. For instance, the conjunctive structure shown in Figure 1 will generate the pattern (1).

(1) [Vc DM] (VH Na Caa Neu Na) [DE Na]

The long pattern has low applicability and hardly

[1] Caa is a PoS for coordinate conjunction. Na is a common noun; Nc denotes place noun, and Vc is a transitive verb. DE denotes the relativizer ‘的’.

can evaluate its precision. Therefore a rule generalization process is applied. Two kinds of generalizations are available. One is reducing the length of contextual constrains and the other is to reduce a fine-grained PoS constraint to a coarse-grained PoS. Some instances, shown in (2), are the generalized patterns of (1).

(2) [DM] (VH Na Caa Neu Na) [DE];1;1
(VH Na Caa Neu Na); 10; 5
[DM] (V N Caa N N) [DE]; 3; 2

Then the applicability and precision of rules higher than threshold values will be selected. The threshold values for the rule selection are determined by testing results on the development data.

3 Resolution of Complex Conjunctive Structures

Complex structures are cases whose boundaries can not be identified by the pattern matching at phase-1. We propose a divide-and-conquer approach to resolve the problem. An input sentence with complex conjunctive structure was first divided into two parts with each part containing one of the conjuncts and then parsed independently to produce their syntactic structures for detecting possible boundaries of two conjuncts. Then ambiguous candidate structures are generated and the best conjunctive structure is selected by evaluating syntactic symmetry and semantic reasonableness of the candidates. Since the two parts of the partial sentences are simple without conjunctive structure and normally grammatical [2], hence they can be easily parsed by a PCFG parser.

Here we illustrate the divide-and-conquer algorithm by the following example. For instance, the example shown in Figure 1 has complex conjunctive structure and it was first split into two parts (1a) and (1b) at conjunction marker "、".

(1a) 如果 *if* (Cbb) 我 *I* (Nh) 發明 *invent* (VC) 一種 *a kind* (DM) 低 *low* (VH) 汙染 *pollution* (Na)

(1b) 零 *null* (Neu) 車禍 *accident* (Na) 的(DE) 汽車 *car* (Na)

The two parts of partial sentences are then parsed to produce their syntactic structures as shown in Figure 1. Then a CRF model trained from Sinica Treebank for checking syntactic symmetry was derived to pick the top-N candidates according to the syntactic information of both sides of partial sentences. Then at the second stage, a semantic evaluation model is proposed to select the best candidate. The detail of the semantic evaluation model is described in the section 3.2. The reason for using a two-stage approach is that the size of the Treebank is limited, but the semantic evaluation model requires the values of association strengths between words. The current Treebank cannot provide enough coverage and reliable values of word-association strengths.

3.1 Derive and evaluate possible candidates

CRF is a well-known probabilistic framework for segmenting and labeling sequence data (Lafferty, et al. 2001). In our experiments, we regard the problem of boundary detection as a chunking-like problem (Lee et al., 2005). Due to this reason, we use CRF model to generate candidates and their ranks. The features used in CRF model included some global syntactic information, such as syntactic category of a partial structure and its phrasal head. Such global syntactic information is crucial for the success of boundary detection and is not available if without the step of parsing process.

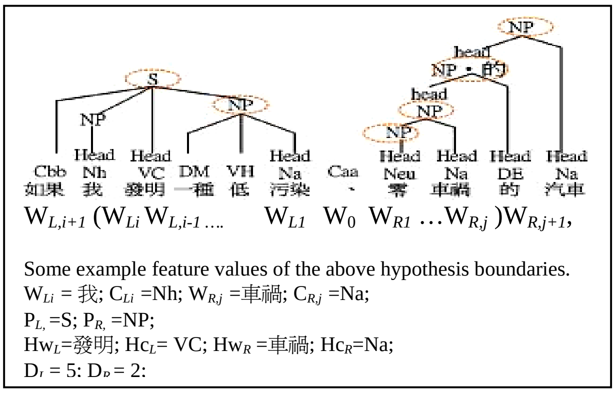


Figure 1. The syntactic structures of 5(a) and 5(b) produced by a PCFG parser.

The features used are:

$W_{L,i}$; $C_{L,i}$; $W_{R,j}$; $C_{R,j}$: The left(*i*)/right(*j*) most word and its pos category of the left/right conjunct.

$P_{L,}$; $P_{R,}$: The phrasal category of the left/right conjunct.

Hw_L ; Hc_L ; Hw_R ; Hc_R: The phrasal head and its pos category of the left/right conjunct.

D_L ; D_R: The length of the left/right conjunct.

Three types of feature patterns are used for CRF. The first type is feature patterns regarding individ-

[2] According to our experiments only 0.8% of the complex testing data and development data are failed to parse their partial structures at both sides of conjunction.

ual conjuncts. The second type is feature patterns regarding symmetry between two conjuncts. The third type is feature patterns regarding contextual properness of a conjunctive structure.

Type1: W_{Li}, W_{Li-1}, W_{Li+1}, C_{Li}, C_{Li-1}, C_{Li-2}, $C_{Li-1}C_{Li-2}$, C_{Li+1}, C_{Li+2}, $C_{Li+1}C_{Li+2}$, $C_{Li}C_{Li-1}C_{Li-2}$, $C_{Li-1}C_{Li}C_{Li+1}$, $C_{Li}C_{Li+1}C_{Li+2}$, $W_{Li}Hw_{L}$, $C_{Li}Hc_{L}$, and W_{Rj}, W_{Rj-1}, W_{Rj+1}, C_{Rj}, C_{Rj-1}, C_{Rj-2}, $C_{Rj-1}C_{Rj-2}$, C_{Rj+1}, C_{Rj+2}, $C_{Rj+1}C_{Rj+2}$, $C_{Rj}C_{Rj-1}C_{Rj-2}$, $C_{Rj-1}C_{Rj}C_{Rj+1}$, $C_{Rj}C_{Rj+1}C_{Rj+2}$, $W_{Rj}Hw_{R}$, $C_{Rj}Hc_{R}$..

Type 2: P_L P_R, Hw_LHw_R, Hc_LHc_R, D_LD_R.

Type 3: $W_{L,i+1}Hw_{Rj}$, $W_{R,j+1}Hw_{Li}$, $W_{L,1}W_{R,j}$, $W_{R,1}W_{L,j}$, $W_{L,1}W_{R,j+1}$, $W_{R,1}W_{L,j+1}$, $W_{L,1}W_{R,j}W_{R,j+1}$, $W_{R,1}W_{L,i}W_{L,i+1}$, $W_{L,1}W_{R,j-1}W_{R,j}$, $W_{R,1}W_{L,i-1}W_{L,i}$, $C_{L,i-1}Hc_{Rj}$, $C_{R,j-1}Hc_{Li}$, $C_{L,i+1}Hc_{Rj}$, $C_{R,j+1}Hc_{Li}$, $C_{L,i}C_{L,i+1}Hc_{Rj}$, $C_{R,j}C_{R,j+1}Hc_{Li}$, $C_{L,1}C_{R,j}$, $C_{R,1}C_{L,j}$, $C_{L,1}C_{R,j+1}$, $C_{R,1}C_{L,j+1}$, $C_{L,1}C_{R,j}C_{R,j+1}$, $C_{R,1}C_{L,i}C_{L,i+1}$, $C_{L,1}C_{R,j-1}C_{R,j}$, $C_{R,1}C_{L,i-1}C_{L,i}$.

A CRF model is trained from the Sinica Treebank and estimated the probabilities of hypothesis conjunctive boundary pairs by the feature patterns listed above. The top ranked candidates are selected according to the CRF model. In general, for further improvement, a final step of semantic evaluation will be performed to select the best candidate from top-N boundary structures ranked by the CRF model, which is described in the next section.

3.2 The word-association evaluation model

For the purpose of selecting the best candidates of complex conjunctive structures, a word association evaluation model is adopted (Hsieh et al. 2007). The word-to-word association data is learned automatically by parsing texts from the Taiwan Central News Agency corpus (traditional characters), which contains 735 million characters. The syntactically dependent words-pairs are extracted from the parsed trees. The word-pairs are phrasal heads and their arguments or modifiers. Though the data is imperfect (due to some errors produced by auto-tagging system and parser), the amount of data is large enough to compensate parsing errors and reliably exhibit strength between two words/concepts.

37,489,408 sentences in CNA (Central News Agency) corpus are successfully parsed and the number of extracted word associations is 221,482,591. The word association probabilities is estimated by eq.(1).

$$P(Modify \mid Head) = \frac{freq(Head, Modify)}{freq(Head)} \qquad (1)$$

"*freq*(*Head*)" means Head word frequency in the corpus and "*freq*(*Head,Modify*)" is the cooccurrence frequency of Head and Modify/Argument.

The final evaluation is done by combining three scores, i.e. (1) the probability produced by PCFG parser, (2) the scores of CRF classifier and (3) the scores of semantic evaluation. The detail is described in Section 4.2.

4 Experiments

3,484 sentences of the Sinica Treebank are used as training data. The development data and testing data are extracted from three different set of corpora the Sinica corpus, Sinorama magazines and textbooks of elementary school (Hsieh et al. 2005). They are totally 202 sentences (244 conjunctions) with 6-10 words and 107 sentences (159 conjunctions) with more than 11 words. We only test the sentences which contain the coordinate conjunction category or categories.

We adopt the standard PARSEVAL metrics (Manning et al., 1999) including bracket f-score to evaluate the performance of the tree structures of sentences and accuracies of boundary detection of conjunction structures.

4.1 Phase-1 experimental results

For the phase-1 experiments, the context-dependent rules are extracted and generalized from Sinica treebank. We then use the development data to evaluate the performances for different sets of rules selected by different threshold values. The results show that the threshold values of occurrence once and precision 70% performed best. This means any context-dependent rule with precision greater than or equal to 70% is used for the future processes. 39941 rules are in the set. In Table 1, we compare *the phase-1* result with *the baseline model* on test data. It is shown that the boundary detection precision is very high, but the recall rate is comparatively low, since *the phase-1* process cannot handle the complex cases. We also compare the processing time between *the baseline model* and *the phase-1 parsing processes* in Table 2. Marking conjunctive boundaries before parsing can limit the search range for parser and save processing time. The effect is more obvious when parsing long sentences. Because long sentences generate more am-

biguous paths than shorter sentences, these surely spend much more time.

Test data	6-10 words		more than 11 words	
	Baseline	*phase1*	Baseline	*phase1*
C-boundary f-score	55.74	84.43	50.0	63.75
S-bracket f-score	72.67	84.44	71.20	79.40

Table 1. The comparison between *the baseline PCFG model* and the *phase1 parsing process* .

unit: second	6-10 words		more than 11 words	
	Baseline	*phase1*	Baseline	*phase1*
development data	14	12	34	23
test data	14	11	34	24

Table 2. The comparison of processing time between *the baseline model* and the *phase1 parsing process*.

4.2 Phase-2 experimental results

Complex cases cannot be matched by context-dependent rules at the phrase-1 which will be handled by the phase-2 algorithms mentioned in Section 3. We use the CRF++ tool (Kudo, 2006) to train our CRF model. The CRF model can produce the N-best candidates for an input conjunctive sentence. We experiment on the models of Top1-CRF and TopN-CRF where the Top1-CRF algorithm means that the final output is the best candidate produced by CRF model and the TopN-CRF means that the final output is the best candidate produced by the structure evaluation process described below.

For each N-best candidate structure, three evaluation scores is derived: (a) the probability score generated from the PCFG parser, i.e. RuleScore, (b) the probability score generated from the CRF classifier, i.e. CRF-Score, and (c) the word association score, i.e. WA-Score. We normalize each of the three scores by eq.(2):

$$normal(Score_i) = \frac{Score_i - Score_{\min}}{Score_{\max} - Score_{\min}} \quad (2)$$

$Score_i$ means the score of the i-th candidate, and $Score_{\min}$ and $Score_{\max}$ mean the worst and the best score in the candidate set for a target conjunctive sentence. The normalized scores are between 0 and 1. After normalization, we combine the three scores with different weights:

Total Score = w1*RuleScore + w2*CRF-Score + w3*WA-Score (3)

The w1, w2 and w3 are regarded as the degree of importance of the three types of information. We use development data to determine the best combination of w1, w2, w3. Due to limit amount of development data, many local maximum and global maximum are achieved by different values of w1, w2, w3. Therefore we use a clustering algorithm to cluster the grid points of (w1, w2, w3) which produce the best performance. We then pick the largest cluster and calculate its centroid as our final weights which are shown at Table 3.

	Top N	w1	w2	w3
6-10words	N = 3	0.11	0.64	0.25
11- words	N = 3	0.18	0.76	0.06

Table 3. The best weights determined by the development data for the sentences with different lengths using the best-3 candidates.

The performance results of the testing data are shown in Table 4. In comparing with the results of the baseline model shown in Table 1, the conjunction boundary f-score increased from about 53% to 83% for the testing data. The processes also improve the overall parsing f-scores from 72% to 83%. The results of Table 4 also show that the evaluation function indeed improves the performances but marginally. However the experiments are done under the condition that the input sentences are perfectly word segmented and pos tagged. In real practices, parser may accept sentences with ambiguous word segmentation and pos tagging to avoid the error accumulation due to early commitment on word segmentation and pos tagging. Therefore parsers require much more information to resolve much more ambiguous conditions. A robust evaluation function may play a very important role. We will do more researches in the future.

		Top1CRF	TopNCRF
Development data	C-boundary f-score	85.57	89.55
	S-bracket f-score	80.10	82.34
Test data	C-boundary f-score	82.18	83.17
	S-bracket f-score	83.15	83.45

Table 4. The final results of our overall processes.

Another point worth mentioning, the performances of “CRF” (using CRF model without phase-1) and “phase1+CRF” (using CRF model after phase-1) algorithms are comparable. However “phase1+ CRF” algorithm is much more efficient, since “phase1+CRF” algorithm can determine the simple conjunctive structures by pattern matching and most of conjunctive structures are simple. On the other hand, the “CRF” model requires twice partial sentence parsing, generates candidates with CRF

classifier and evaluates structure with three syntactic and semantic scores.

5 Conclusion

Conjunctive boundary detection is not a simple task. It is not only time consuming but also knowledge intensive. Therefore we propose a context-dependent rules matching approach to handle simple cases to get fast returns. For complex cases, we use a knowledge intensive divide-and-conquer approach. To resolve the problems of inadequate knowledge and data sparseness due to limit amount of structure annotated training data, we extract word/concept associations from CNA corpus.

In our experiments, the proposed model works well. Most conjunctive phrases are simple cases and can be matched by context-dependent rules and indeed avoid unnecessary calculation. Compared with the baseline method of straight forward PCFG parsing, the f-score of conjunctive boundary detection can be raised about 22%. For the complex cases, the boundaries f-score is further raised about 7% after phase-2 processes. The experimental results show that the method not only works well on boundary resolution for conjunctive phrases but also improves the total performances of syntactic parsing.

Our solutions include the rule-based method and cooperate with semantic and syntactic analyses. Therefore in the future we will try to enhance the syntactic and semantic analyses. For syntactic analysis, we still need to find more effective methods to improve the performance of our parser. For the semantic analysis, we will try to refine the word association data and discover a better semantic evaluation model.

Acknowledgements

This research was supported in part by National Digital Archives Program (NDAP, Taiwan) sponsored by the National Science Council of Taiwan under NSC Grants: NSC95-2422-H-001-031-.

References

Agarwal, Rajeev and Boggess, Lois. 1992. A Simple but Useful Approach to Conjunct Identification. In *Proceedings of 30th Annual Meeting of Association for Computational Linguistics*, pages 15-21.

Chen, Keh-Jiann, Huang, Chu-Ren, Chen, Feng-Yi, Luo, Chi-Ching, Chang, Ming-Chung, Chen, Chao-Jan and Gao, Zhao-Ming. 2003. Sinica Treebank: design criteria, representational issues and implementation. In Anne Abeille, (ed.): *Building and Using Parsed Corpora. Text, Speech and Language Technology.* 20:231-248, pages 231-248.

Hsieh,Yu-Min, Yang, Duen-Chi and Chen, Keh-Jiann. 2005. Linguistically-motivated grammar extraction, generalization and adaptation. In *Proceedings of the Second International Join Conference on Natural Language Processing (IJCNLP2005)*, pages 177-187, Jeju Island, Republic of Korea.

Hsieh, Yu-Ming, Duen-Chi Yang and Keh-Jiann Chen. 2007. Improve Parsing Performance by Self-Learning. *International Journal of Computational Linguistics and Chinese Language Processing*, Vol. 12, #2, pages 195-216.

Kurohashi, Sadao, and Nagao, Makoto. 1994. A Syntactic Analysis Method of Long Japanese Sentences Based on the Detection of Conjunctive Structure. Computational Linguistics 20(4), pages 507-534.

Kudo, Taku. 2006. (software)CRF++: Yet Another CRF toolkit *http://chasen.org/~taku/software/CRF++/.*

Lafferty, John, McCallum, Andrew, Pereira, Fernando. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning* (*ICML-01*), pages 282-289.

Lee, Yong-Hun, Kim, Mi-Young and Lee, Jong-Hyeok. 2005. Chunking Using Conditional Random Fields in Korea Texts. In *Proceedings of the Second International Join Conference on Natural Language Processing (IJCNLP2005)*, pages 155-164, Jeju Island, Republic of Korea.

Manning, Christopher D., and Schutze, Hinrich. 1999. Foundations of Statistical Natural Language processing. *The MIT Press*, Cambridge, Massachusetts.

Miller, Geroge, 1993. Introduction to WordNet: An Online Lexical Database. Princeton, CSL Report 43.

Steiner, Ilona. 2003. Parsing Syntactic Redundancies in Coordinate Structures. Poster presentation at the *European Cognitive Science Conference (EuroCogSci03).*

Van Delden, Sebastian. 2002. A Hybrid Approach to Pre-Conjunct Identification. In *Proceedings of the 2002 Language Engineering Conference* (LEC 2002), pages 72-77, University of Hyderabad, India.

Wu, Yunfang. 2003. Contextual Information of Coordinate Structure. Advances on the Research of Machine Translation, pages 103-109, Publishing house of Electronics Industry.

Dependency Annotation Scheme for Indian Languages

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai and Rajeev Sangal
Language Technologies Research Center,
IIIT, Hyderabad, India.
{rafiya,samar}@research.iiit.ac.in,
{dipti,lakshmi,sangal}@iiit.ac.in

Abstract

The paper introduces a dependency annotation effort which aims to fully annotate a million word Hindi corpus. It is the first attempt of its kind to develop a large scale tree-bank for an Indian language. In this paper we provide the motivation for following the Paninian framework as the annotation scheme and argue that the Paninian framework is better suited to model the various linguistic phenomena manifest in Indian languages. We present the basic annotation scheme. We also show how the scheme handles some phenomenon such as complex verbs, ellipses, etc. Empirical results of some experiments done on the currently annotated sentences are also reported.

1 Introduction

A major effort is currently underway to develop a large scale tree bank for Indian Languages (IL). The lack of such a resource has been a major limiting factor in the development of good natural language tools and applications for ILs. Apart from that, a rich and large-scale tree bank can be an indispensable resource for linguistic investigations. Some notable efforts in this direction for other languages have been the Penn Tree Bank (Marcus et al., 1993) for English and the Prague Dependency Bank (Hajicova, 1998) for Czech.

It is well known that context free grammar (CFG) is not well-suited for free-word order languages (Shieber, 1985); instead dependency framework appears to be better suited (Hudson, 1984; Mel'Cuk, 1988, Bharati et al., 1995). Also, the dependency framework is arguably closer to semantics than the phrase structure grammar (PSG) if the dependency relations are judiciously chosen. In recent times many research groups have been shifting to the dependency paradigm due to this reason. Modern dependency grammar is attributed to Tesnière (1959). In a dependency analysis, there is no hierarchical arrangement of phrases (or substrings) like in phrase structure grammar. Rather, we just have words connected via dependency relations between them.

Prague Dependency Bank (PDT) for Czech (which has relatively free word order) is one such large-scale effort which implements a three-tier annotation scheme and annotates morphological information, analytical and tectogrammatical level annotations at these three levels. Out of the three levels, the analytical and tectogrammatical level are dependency based. The tectogrammatical level tries to capture the deep-semantics of the sentence; the annotation at this level is very rich and is linked to the other two lower levels. Other major efforts in the dependency framework are Alpino (van der Beek et. al, 2002) for Dutch, (Rambow et. al, 2002) for English, TUT (Bosco and Lombardo, 2004) for Italian, TIGER (Brants et. al, 2002) (combines dependency with PSG) for German. In this paper we describe an approach to annotate ILs using the Paninian[1] model. The paper is arranged as follows, Section 2 gives a brief overview of the

[1] Paninian theory was formulated by Panini about two thousand five hundred years ago for Sanskrit. It evolved with the contributions of grammarians that followed.

grammatical model and the motivation for following the framework. Section 3 talks about the chosen corpus and the annotation procedure. In Section 4 we discuss some dependency relations. Section 5 describes the evaluation procedure. We report the empirical results of experiments done on the annotated data in Section 6. Section 7, concludes the paper.

2 Grammatical Model

ILs are morphologically rich and have a relatively flexible word order. For such languages syntactic subject-object positions are not always able to elegantly explain the varied linguistic phenomena. In fact, there is a debate in the literature whether the notions 'subject' and 'object' can at all be defined for ILs (Mohanan, 1982). Behavioral properties are the only criteria based on which one can confidently identify grammatical functions in Hindi (Mohanan, 1994); it can be difficult to exploit such properties computationally. Marking semantic properties such as thematic role as dependency relation is also problematic. Thematic roles are abstract notions and will require higher semantic features which are difficult to formulate and to extract as well. So, thematic roles are not marked at this juncture. On the other hand, the notion of *karaka* relations (explained shortly) provides us a level which while being syntactically grounded also helps in capturing some semantics. What is important to note here is that such a level can be exploited computationally with ease. This provides us with just the right level of syntactico-semantic interface. The experiments conducted on the present annotated text provide empirical evidence for this claim (section 6). Paninian grammar is basically a dependency grammar (Kiparsky and Staal, 1969; Shastri, 1973). In this section we briefly discuss the Paninian model for ILs and lay down some basic concepts inherent to this framework.

The main problem that the Paninian approach addresses is to identify syntactico-semantic relations in a sentence. The Paninian approach treats a sentence as a series of modifier-modified relations. A sentence is supposed to have a primary modified (the root of the dependency tree) which is generally the main verb of the sentence. The elements modifying the verb participate in the action specified by the verb. The participant relations with the verb are called *karaka*. The appropriate mapping of the syntactic cues helps in identifying the appropriate karakas ('participants in an action'). The framework is inspired by an inflectionally rich language like Sanskrit; it emphasizes the role of case endings or markers such as post-positions and verbal inflections.

There are six basic karakas, namely; *adhikarana* 'location', *apaadaan* 'source', *sampradaan* 'recipient', *karana* 'instrument', *karma* 'theme', *karta* 'agent'. We must note here that although one can roughly map the first four karaka to their thematic role counterpart, karma and karta are very different from 'theme' and 'agent' respectively (see section 4.1.1).

In our annotation scheme, we use chunks as a device for modularity. A chunk represents a set of adjacent words which are in dependency relations with each other, and are connected to the rest of the words by a single incoming dependency arc. The relations among the words in a chunk are not marked for now and hence allow us to ignore local details while building the sentence level dependency tree. Thus, in our dependency tree each node is a chunk and the edge represents the relations between the connected nodes labeled with the karaka or other relations. All the modifier-modified relations between the heads of the chunks (inter-chunk relations) are marked in this manner. Intra-chunk relations can be marked by a set of rules at a later point. Experiments have been conducted with high performance in automatically marking intra-chunk dependencies.

Using information such as karakas based on some vibhaktis (post-positions) and other information like TAM (tense, aspect and modality) of the main verb seems very well suited for handling free word order languages. Other works based on this scheme like (Bharati et al., 1993; Bharati et al., 2002; Pedersen et al., 2004) have shown promising results. We, therefore, propose the use of dependency annotation based on the Paninian model in the Indian context.

3 Annotation Procedure and Corpus Description

The annotation task is planned on a million word Hindi corpora obtained from CIIL (Central Institute for Indian Languages), Mysore, India. It is a representative corpus which contains texts from various domains like newspaper, literature, gov-

ernment reports, etc. The present subset on which the dependency annotation is being performed has already been manually tagged and chunked. Currently the annotation is being carried out by 2 annotators, who are graduate students with linguistic knowledge. The tool being used for the annotation is part of *Sanchay* (Singh, 2006) which is a collection of tools and APIs for South Asian languages.

4 Scheme

There are a total of 28 relations (see, *http://ltrc.deptagset.googlepages.com/home*) which we encode during the annotation. The total number of relations in the framework is few which has a direct bearing on the parser based on this framework (both, rule based and statistical). We briefly discuss some of these relations in this section.

4.1 Dependency relations

As mentioned earlier, the proposed scheme uses the dependency relations from Paninian grammar. Section 4.1.1 below shows some karaka relations, section 4.1.2 shows some other relations;

4.1.1 *karaka* relations

(1) *raama phala khaataa hai*
'Ram' 'fruit' 'eat' 'is'
'Ram eats fruit'

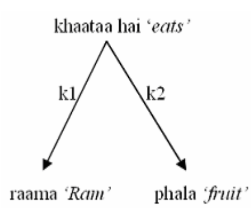


Figure1.

(2) *raama chaaku se saiv kaattaa hai*
'Ram' 'knife' *-inst* 'apple' 'cut' 'is'
'Ram cuts the apple with a knife'

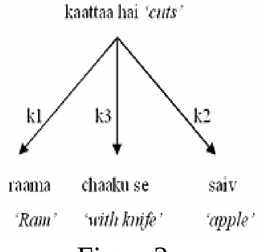


Figure2.

Examples (1), and (2) above show some simple cases which have *karaka* relations such as k3 (karana; 'instrument'), k1 (karta), k2 (karma) (the term karta and karma can be roughly translated as 'agent' and 'theme'). One must note here that the notion of karta, karma, etc, is not equivalent to that of the 'agent', 'theme' thematic roles (although they might map to them sometimes). The reason for this divergence in the two notions (karaka and thematic role) is due to the difference in what they convey. Thematic role is purely semantic in nature whereas the karaka is syntactico-semantic. Examples (3), illustrates this point,

(3) chaabhi ne darvaazaa kholaa
'key' '*-erg*' '*door*' '*opened*'
'The key opened the door'

In the above examples *chaabhi* is k1 (*karta*), whereas it takes instrument thematic role. While the karaka relations are primarily motivated via verbal semantics, syntactic cues like postpositions and verbal morphology play an important role too. For example in (3) above, the ergative case '*ne*' provides a strong cue to identify *karta*. Panini defines *'karta'* as *'svatantra karta'* which can be translated as 'the participant which is the most independent in a given action'. In (3) 'key' has such a property. When the speaker uses 'key' in (3), he/she intends to elevate the role of 'key' in the action of opening and does not communicate the actual agent of the action. The speaker uses 'key' as the independent participant in the act of opening. Hence, 'key' is the *karta* (see Bharati et al., 1995, pp. 65-73, for a more detailed discussion).

4.2 Special Cases

(a) **POF (Part of relation)**

Conjunct verbs form a very challenging case for analysis in Indian languages. They have been extensively analyzed in the past. Some notable attempts have been (Greaves, 1983; Kellogg, 1972; Mohanan, 1994; Butt, 2004). The example below shows a N+V conjunct verb usage;

(4) *raama ne mujhase prashna kiyaa*
'Ram' *-erg* 'me-*inst*' 'question' 'do'
'Ram asked me a question.'

In example (4), *prashna kiyaa* is a conjunct verb and behaves as a single semantic unit. These verbs can also be discontiguous as in (5),

(5) *raama ne mujhase prashna pichle saal kiyaa*
'Ram' *-erg* 'me-*inst*' 'question' 'last' 'year' 'did'
'Ram asked me a question last year.'

In the above example above a normal conjunct verb sequence *prashna kiyaa* is disjoint, making it rather difficult to annotate. In fact, practically anything can come between the disjointed elements. Ideally, the noun/adjective + verb sequence of the conjunct verb is placed in one chunk. Keeping this in mind, example (6) below is even more problematic,

(6) *maine usase ek prashna kiyaa*
'I-*erg*' 'him-*inst*' 'one' 'question' 'did'
'I asked him a question'

The noun *prashna* 'question' within the conjunct verb sequence *prashna kiyaa* is being modified by the adjective *ek* 'one' and not the entire noun-verb sequence; the annotation scheme should be able to account for this relation in the dependency tree. If *prashna kiyaa* is grouped as a single verb chunk, it will not be possible to mark the appropriate relation between *ek* and *prashna*. To overcome this problem it is proposed to break *ek prashna kiyaa* into two separate chunks, [*ek prashna*]/NP[2] [*kiyaa*]/VG[3]. The dependency relation of *prashna* with *kiyaa* will be **POF** ('Part OF' relation), i.e. the noun or an adjective in the conjunct verb sequence will have a POF relation with the verb. This way, the relation between *ek* and *prashna* becomes an intra-chunk relation as they will now become part of a single NP chunk. What makes such a sequence unique is the fact that the components which make up a conjunct verb are chunked separately, but semantically they constitute a single unit.

The proposed scheme has the following advantages:

(i) It captures the fact that the noun-verb seque ence is a conjunct verb by linking them with an appropriate tag, this information is extremely crucial syntactically.

(ii) It allows us to deal with the modifier-modified relation between an adjective and its modified noun, as in example (6), which is a frequent phenomenon.

The tree below shows the proposed solution, where the adjective *ek* modifies the noun *prashna* instead of the entire *prashna kiyaa*, which would have been the case had we not separated *prashna kiyaa* into two separate chunks.

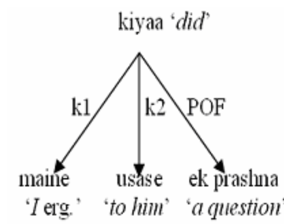


Figure3.

(b) **ccof ('conjunct of' relation) and ellipses**

In the case of coordinating conjunction like *aur* 'and' , the conjunct becomes the root and takes the two conjoined elements as children, the relation marked on the edges is **ccof** (conjunct of). This analysis captures the fact that neither of the conjoined elements is the head. (The head of the two (or more) conjoined elements lies in the conjunct, and may be so computed when needed.) The elements participating in the coordination can belong to various categories, such as, noun, adjective, adverbs etc; they can also be entire clauses, participles, etc. Other conjunct and punctuations which act like conjuncts are annotated similarly.

When one or more element from a sentence is dropped, it is called ellipses. A null element marked with a special tag 'NULL' is introduced in cases of ellipses, where without inserting it the tree cannot be drawn. Null_NP, Null_VG, Null_CCP etc mark different kinds of ellipses.

In this section, we have briefly discussed some of the relations and showed their actual usage using some sentences. The number of tags in the proposed scheme is not very large. A limited set of tags helps immensely in developing high-performance parsers (both rule based and statistical) and other related applications. We should note here that our tag-set, although small, is not a de-

[2] Noun Phrase
[3] Verb Group

limiting factor and is not a compromise on the semantics, as these 28 relations are enough to fully parse the sentences in the language.

5 Evaluation

To make sure that the quality of the annotated corpus is good, the annotators cross-validate each other's work. A senior team member finally checks the annotated corpus (of both the annotators) to ensure that the errors are minimized. Note that such a setup is only temporary, we need such a thorough validation because we are still in the process of revising the guidelines. Once the guidelines become stable, the annotators won't need to cross-validate. Of course, the task of final validation will still continue.

6 Experiments

Some preliminary experiments were conducted on a corpus of 1403 Hindi sentences that have been fully annotated. The aim was to access;

1. Whether the syntactic cues can be exploited for better machine learnability.
2. Whether certain generalization can be made for a constraint parser.
3. How far would the automatic annotation help the annotators?

We found a strong co-relation between most vibhakti-karaka occurrences (shaded cells in Table 1). k7 ('place') for example, overwhelmingly takes *mem* post-position, k3 (*karana*) takes *se* in all the cases. Of course, there are some competing relations which show preference for the same post-position. In such cases only the post-position information will not be sufficient and we need to take into account other syntactic cues as well. These syntactic cues can be TAM (tense, aspect and modality) of the verb, verb class information, etc. For example, in case of *karata karaka* (k1), the following heuristics help resolve the ambiguities seen in Table 1. These heuristics are applied sequentially, i.e. if the first fails then the next follows. Note that the heuristics mentioned below are meant only for illustrative purpose. The cues mentioned in the heuristics will finally be used as features by an efficient ML technique to automate the task of annotation.

H1: k1 agrees in gender, number and person with the verb if it takes a nominative case,
H2: k1 takes a *ko* case-marker if the TAM of the verb has *nA*,
H3: It takes a *kaa/ke/ki* if the verb is infinitive,
H4: It takes a *se* or *dvaara* if the TAM of the verb is passive
H5: It takes a *ne* case-marker if the verb is transitive and the TAM is perfective

Table-2 shows the results when the heuristics were tested on the annotated corpus to test their effectiveness.

V[5] / Kr[4]	kA	se	par	mem	ne	ko
k1	53	5	0	0	344	127
k2	174	70	5	5	0	280
k3	0	51	0	0	0	0
k4	0	9	0	0	0	77
k5	1	97	0	0	0	0
k7	1	4	38	141	0	0
k7p	45	11	72	302	0	0
k7t	13	12	6	31	0	7

Table 1. karaka-vibhakti correlation

	Total	Correct Identified	%
H1	1801	1461	81.1
H2	127	35	27.5
H3	53	19	35.1
H4	10	10	100
H5	344	330	95.9

Table 2. Heuristics for k1 disambiguation

The field 'Total' in Table-2 gives us the number of instances where a particular heuristic was applied. For example, there were 1801 instances where k1 appeared in a nominative case and H1 correctly identified 1461 instances. H1 failed due to the errors caused by the morphological analyzer, presence of conjuncts, etc. Of particular interest are H2 and H3 which didn't work out for large number of cases. It turns out that H2 failed for what is understood in the literature as dative subjects. Dative subjects occur with some specific verbs, one possible solution could be to use such verbs for disambiguation. Automatic identification of conjunct verbs is a difficult problem; in fact, there isn't any robust linguistic test which can be

[4] karaka relations (see, http://ltrc.deptagset.googlepages.com/home)
[5] vibhakti (post-position)

used to identify such verbs. Similar heuristics can be proposed for disambiguating other karaka based on some syntactic cues. Based on the above results one can safely conclude that arriving at some robust generalization (like, karaka-vibhakti correlation) based on the syntactic cues is in fact possible. This can help us immensely in building an efficient parser for Hindi (and other ILs). It goes without saying that there exists a lot of scope for automating the annotation task.

7 Conclusion

In this paper we have introduced an ongoing effort to annotate Indian languages with dependency relation. We stated the motivation behind following the Paninian framework in the Indian Language scenario. We discussed the basic scheme along with some new relations such as ccof, POF, etc. We also showed the results of some experiments conducted on the annotated data which showed that there is a strong co-relation between vibhakti-karaka relations.

Acknowledgement

Thanks are due to Prof. Ramakrishnamacharyulu who has been guiding us throughout the development of the proposed scheme.

References

Akshar Bharati and Rajeev Sangal. 1993. Parsing Free Word Order Languages in the Paninian Framework, *ACL93: Proc. of Annual Meeting of Association for Computational Linguistics*.

Akshar Bharati, Vineet Chaitanya and Rajeev Sangal. 1995. *Natural Language Processing: A Paninian Perspective*, Prentice-Hall of India, New Delhi, pp. 65-106.

Akshar Bharati, Rajeev Sangal, T Papi Reddy. 2002. A Constraint Based Parser Using Integer Programming, In *Proc. of ICON-2002: International Conference on Natural Language Processing, 2002*.

Cristina Bosco and V. Lombardo. 2004. Dependency and relational structure in treebank annotation. In *Proceedings of Workshop on Recent Advances in Dependency Grammar at COLING'04*.

S. Brants, S. Dipper, S. Hansen, W. Lezius and G. Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*.

M. Butt. 2004. The Light Verb Jungle. In G. Aygen, C. Bowern & C. Quinn eds. *Papers from the GSAS/Dudley House Workshop on Light Verbs.* Cambridge, Harvard Working Papers in Linguistics, p. 1-50.

Edwin Greaves. 1983. *Hindi Grammar*. Asian Educational Services, New Delhi, pp. 335-340

E. Hajicova. 1998. Prague Dependency Treebank: From Analytic to Tectogrammatical Annotation. In *Proc. TSD'98*.

R. Hudson. 1984. *Word Grammar*, Basil Blackwell, 108 Cowley Rd, Oxford, OX4 1JF, England.

S. H. Kellogg. 1972. *A Grammar of the Hindi Language.* Munshiram Manoharlal, New Delhi, pp. 271-279.

P. Kiparsky and J. F. Staal. 1969. 'Syntactic and Relations in Panini', *Foundations of Language* 5, 84-117.

M. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank, *Computational Linguistics 1993*.

I. A. Mel'cuk. 1988. *Dependency Syntax: Theory and Practice*, State University, Press of New York.

K. P. Mohanan. 1982. Grammatical relations in Malayalam, *In* Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge.

Tara Mohanan, 1994. *Arguments in Hindi*. CSLI Publications.

M. Pedersen, D. Eades, S. K. Amin, and L. Prakash. 2004. Relative Clauses in Hindi and Arabic: A Paninian Dependency Grammar Analysis. In *COLING 2004 Recent Advances in Dependency Grammar*, pages 9–16. Geneva, Switzerland.

O. Rambow, C. Creswell, R. Szekely, H. Taber, and M. Walker. 2002. A dependency treebank for English. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation.*

Anil Kumar Singh. 2006. *http://sourceforge.net/projects/nlp-sanchay*

Charudev Shastri. 1973. *Vyakarana Chandrodya (Vol. 1 to 5)*. Delhi: Motilal Banarsidass. (In Hindi)

S. M. Shieber. 1985. Evidence against the context-freeness of natural language. In *Linguistics and Philosophy*, p. 8, 334–343.

L. Tesnière. 1959. *Eléments de Syntaxe Structurale*. Klincksiek, Paris.

L. van der Beek, G. Bouma, R. Malouf, and G. van Noord. 2002. The Alpino dependency treebank. *Computational Linguistics in the Netherlands*.

Non-Factoid Japanese Question Answering through Passage Retrieval that Is Weighted Based on Types of Answers

Masaki Murata and Sachiyo Tsukawaki
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
{murata,tsuka}@nict.go.jp

Toshiyuki Kanamaru
Kyoto University
Yoshida-Nihonmatsu-Cho, Sakyo
Kyoto, 606-8501 Japan
kanamaru@hi.h.kyoto-u.ac.jp

Qing Ma
Ryukoku University
Otsu, Shiga, 520-2194, Japan
qma@math.ryukoku.ac.jp

Hitoshi Isahara
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
isahara@nict.go.jp

Abstract

We constructed a system for answering non-factoid Japanese questions. We used various methods of passage retrieval for the system. We extracted paragraphs based on terms from an input question and output them as the preferred answers. We classified the non-factoid questions into six categories. We used a particular method for each category. For example, we increased the scores of paragraphs including the word “reason” for questions including the word “why.” We participated at NTCIR-6 QAC-4, where our system obtained the most correct answers out of all the eight participating teams. The rate of accuracy was 0.77, which indicates that our methods were effective.

1 Introduction

A question-answering system is an application designed to produce the correct answer to a question given as input. For example, when “What is the capital of Japan?” is given as input, a question-answering system may retrieve text containing sentences like “Tokyo is Japan’s capital and the country’s largest and most important city”, and “Tokyo is also one of Japan’s 47 prefectures”, from Websites, newspaper articles, or encyclopedias. The system then outputs “Tokyo” as the correct answer. We believe question-answering systems will become a more convenient alternative to other systems designed for information retrieval and a basic component of future artificial intelligence systems. Numerous researchers have recently been attracted to this important topic. These researchers have produced many interesting studies on question-answering systems (Kupiec, 1993; Ittycheriah et al., 2001; Clarke et al., 2001; Dumis et al., 2002; Magnini et al., 2002; Moldovan et al., 2003). Evaluation conferences and contests on question-answering systems have also been held. In particular, the U.S.A. has held the Text REtrieval Conferences (TREC) (TREC-10 committee, 2001), and Japan has hosted the Question-Answering Challenges (QAC) (National Institute of Informatics, 2002) at NTCIR (NII Test Collection for IR Systems) 3. These conferences and contests have aimed at improving question-answering systems. The researchers who participate in these create question-answering systems that they then use to answer the same questions, and each system’s performance is then evaluated to yield possible improvements.

We addressed non-factoid question answering in NTCIR-6 QAC-4. For example, when the question was “Why are people opposed to the Private Information Protection Law?” the system retrieved sentences based on terms appearing in the question and output an answer using the retrieved sentences. Numerous studies have addressed issues that are involved in the answering of non-factoid questions (Berger et al., 2000; Blair-Goldensohn et al., 2003;

Xu et al., 2003; Soricut and Brill, 2004; Han et al., 2005; Morooka and Fukumoto, 2006; Maehara et al., 2006; Asada, 2006).

We constructed a system for answering non-factoid Japanese questions for QAC-4. We used methods of passage retrieval for the system. We extracted paragraphs based on terms from an input question and output them as the preferred answers. We classified the non-factoid questions into six categories. We used a particular method for each category. For example, we increased the scores of paragraphs including the word "reason" for questions including the word "why." We performed experiments using the NTCIR-6 QAC-4 data collection and tested the effectiveness of our methods.

2 Categories of Non-Factoid Questions

We used six categories of non-factoid questions in this study. We constructed the categories by consulting the dry run data in QAC-4.

1. Definition-oriented questions (Questions that require a definition to be given in response.)

   e.g., *K-1 to wa nandesuka?* (What is K-1?)

2. Reason-oriented questions (Questions that require a reason to be given in response.)

   e.g., *kojin jouhou hokogou ni hantai shiteiru hito wa doushite hantai shiteiru no desuka?* (Why are people opposed to the Private Information Protection Law?)

3. Method-oriented questions (Questions that require an explanation of a method to be given in response.)

   e.g., *sekai isan wa donoyouni shite kimeru no desuka?*" (How is a World Heritage Site determined?)

4. Degree-oriented questions (Questions that require an explanation of the degree of something to be given in response.)

5. Change-oriented questions (Questions that require a description of things that change to be given in response.)

   e.g., *shounen hou wa dou kawari mashitaka?* (How was the juvenile law changed?)

6. Detail-oriented questions (Questions that require a description of the particulars or details surrounding a sequence of events to be given in response.)

   e.g., *donoyouna keii de ryuukyuu oukoku wa nihon no ichibu ni natta no desuka?* (How did Ryukyu come to belong to Japan?)

3 Question-answering Systems in this Study

The system has three basic components:

1. <u>Prediction of type of answer</u>

   The system predicts the answer to be a particular type of expression based on whether the input question is indicated by an interrogative pronoun, an adjective, or an adverb. For example, if the input question is "Why are people opposed to the Private Information Protection Law?", the word "why" suggests that the answer will be an expression that describes a reason.

2. <u>Document retrieval</u>

   The system extracts terms from the input question and retrieves documents by using these terms. Documents that are likely to contain the correct answer are thus gathered during the retrieval process. For example, for the input question "Why are people opposed to the Private Information Protection Law?", the system extracts "people," "opposed," "Private," "Information," "Protection," and "Law" as terms and retrieves the appropriate documents based on these.

3. <u>Answer detection</u>

   The system separates the retrieved documents into paragraphs and retrieves those that contain terms from the input question and a clue expression (e.g., "to wa" (copula sentence) for the definition sentence). The system outputs the retrieved paragraphs as the preferred answer.

3.1 Prediction of type of answer

We used the following rules for predicting the type of answer. We constructed the rules by consulting the dry run data in QAC-4.

1. Definition-oriented questions Questions including expressions such as "*to wa nani*," "*donna*," "*douiu*," "*douitta*," "*nanimono*," "*donoyouna mono*," "*donna mono*," and "*douiu koto*" (which all mean "what is") are recognized by the system as being definition-oriented questions.

2. Reason-oriented questions Questions including expressions such as "*naze*" (why), "*naniyue*" (why), "*doushite*" (why), "*nani ga riyuu de*" (what is the reason), and "*donna riyuu de*" (what reason), are recognized by the system as being reason-oriented questions.

3. Method-oriented questions Questions including expressions such as "*dou*," "*dousureba*," "*douyatte*," "*dono youni shite*," "*ikani shite*," "*ikani*," and "*donnna houhou de*" (which all mean "how") are recognized by the system as being method-oriented questions.

4. Degree-oriented questions Questions including expressions such as "*dorekurai*" (how much), "*dorekurai no*" (to what extent), and "*dono teido*" (to what extent), are recognized by the system as being degree-oriented questions.

5. Change-oriented questions Questions including expressions such as "*naniga chigau*" (What is different), "*donoyuni kawaru*" (How is ... changed), and "*dokoga kotonaru*" (What is different), are recognized by the system as being change-oriented questions.

6. Detail-oriented questions Questions including expressions such as "*dono you na keii*," "*dono you na ikisatsu*," and "*dono you na nariyuki*" (which all mean "how was") are recognized by the system as being detail-oriented questions.

3.2 Document retrieval

Our system extracts terms from a question by using the morphological analyzer, ChaSen (Matsumoto et al., 1999). The analyzer first eliminates prepositions, articles, and similar parts of speech. It then retrieves documents by using the extracted terms.

The documents are retrieved as follows:

We first retrieve the top k_{dr1} documents with the highest scores calculated using the equation

$$Score(d) = \sum_{\text{term } t} \left(\frac{tf(d,t)}{tf(d,t) + k_t \dfrac{length(d) + k_+}{\Delta + k_+}} \times log \frac{N}{df(t)} \right), \quad (1)$$

where d is a document, t is a term extracted from a question, and $tf(d,t)$ is the frequency of t occurring in d. Here, $df(t)$ is the number of documents in which t appears, N is the total number of documents, $length(d)$ is the length of d, and Δ is the average length of all documents. Constants k_t and k_+ are defined based on experimental results. We based this equation on Robertson's equation (Robertson and Walker, 1994; Robertson et al., 1994). This approach is very effective, and we have used it extensively for information retrieval (Murata et al., 2000; Murata et al., 2001; Murata et al., 2002). The question-answering system uses a large number for k_t.

We extracted the top 300 documents and used them in the next procedure.

3.3 Answer detection

In detecting answers, our system first generates candidate expressions for them from the extracted documents. We use two methods for extracting candidate expressions. Method 1 uses a paragraph as a candidate expression. Method 2 uses a paragraph, two continuous paragraphs, or three continuous paragraphs as candidate expressions.

We award each candidate expression the following score.

$$\begin{aligned} &Score(d) \\ = \;& -min_{t1 \in T} log \prod_{t2 \in T3} (2dist(t1,t2) \frac{df(t2)}{N}) \\ + \;& 0.00000001 \times length(d) \\ = \;& max_{t1 \in T} \sum_{t2 \in T3} log \frac{N}{2dist(t1,t2) * df(t2)} \\ + \;& 0.00000001 \times length(d) \end{aligned} \quad (2)$$

$$T3 = \{t | t \in T, 2dist(t1, t) \frac{df(t)}{N} \leq 1\}, \quad (3)$$

where d is a candidate expression, T is the set of terms in the question, $dist(t1, t2)$ is the distance between $t1$ and $t2$ (defined as the number of characters between them with $dist(t1, t2)$ = 0.5 when $t1 = t2$), and $length(d)$ is the number of characters in a candidate expression. The numerical term, $0.00000001 \times length(d)$, is used for increasing the scores of long paragraphs.

For reason-oriented questions, our system uses some reason terms such as "riyuu" (reason), "gen'in" (cause), and "nazenara" (because) as terms for Eq. 2 in addition to terms from the input question. This is because we would like to increase the score of a document that includes reason terms for reason-oriented questions.

For method-oriented questions, our system uses some method terms such as "houhou" (method), "tejun" (procedure), and "kotoniyori" (by doing) as terms for second document retrieval (re-ranking) in addition to terms from the input question.

For detail-oriented questions, our system uses some method terms such as "keii" (a detail, or a sequence of events), "haikei" (background), and "rekishi" (history) as terms for second document retrieval (re-ranking) in addition to terms from the input question.

For degree-oriented questions, when candidate paragraphs include numerical expressions, the score ($Score(d)$) is multiplied by 1.1.

For definition-oriented questions, the system first extracts focus expressions. When the question includes expressions such as "*X-wa*", "*X-towa*", "*X-toiunowa*", and "*X-tte*", X is extracted as a focus expression. The system multiplies the score, ($Score(d)$), of the candidate paragraph having "*X-wa*", "*X-towa* or something by 1.1. When the candidate expression includes focus expressions having modifiers (including modifier clauses and modifier phrases), the modifiers are used as candidate expressions, and the scores of the candidate expressions are multiplied by 1.1.

Below is an example of a candidate expression that is a modifier clause in a sentence.

Table 1: Results

Method	Correct	A	B	C	D
Method 1	57	18	42	10	89
Method 2	77	5	67	19	90

(There were a total of 100 questions.)

Question sentence: *sekai isan jouyaku to wa dono youna jouyaku desu ka?*

(What is the Convention concerning the Protection of the World Cultural and Natural Heritage?)

Sentence including answers:

1972 nen no dai 17 kai yunesuko soukai de saitaku sareta sekai isan jouyaku ....

(Convention concerning the Protection of the World Cultural and Natural Heritage, which was adopted in 1972 in the 17th general assembly meeting of the UN Educational, Scientific and Cultural Organization.)

Finally, our system extracts candidate expressions having high scores, ($Score(d)$s), as the preferred output. Our system extracts candidate expressions having scores that are no less than the highest score multiplied by 0.9 as the preferred output.

We constructed the methods for answer detection by consulting the dry run data in QAC-4.

4 Experiments

The experimental results are listed in Table 1. One hundred non-factoid questions were used in the experiment. The questions, which were generated by the QAC-4 organizers, were natural and not generated by using target documents. The QAC-4 organizers checked four or fewer outputs for each question. Methods 1 and 2 were used to determine what we used as answer candidate expressions (Method 1 uses one paragraph as a candidate answer. Method 2 uses one paragraph, two paragraphs, or three paragraphs as candidate answers.).

"A," "B," "C," and "D" are the evaluation criteria. "A" indicates output that describes the same content as that in the answer. Even if there is a supplementary expression in the output, which does not change

the content, the output is judged to be "A." "B" indicates output that contains some content similar to that in the answer but contains different overall content. "C" indicates output that contains part of the same content as that in the answer. "D" indicates output does not contain any of the same content as that in the answer. The numbers for "A," "B," "C," and "D" in Table 1 indicate the number of questions where an output belongs to "A," "B," "C," and "D". "Correct" indicates the number of questions where an output belongs to "A," "B," or "C". The evaluation criteria "Correct" was also used officially at NTCIR-6 QAC-4.

We found the following.

- Method 1 obtained higher scores in evaluation A than Method 2. This indicates that Method 1 can extract a completely relevant answer more accurately than Method 2.
- Method 2 obtained higher scores in evaluation "Correct" than Method 1. The rate of accuracy for Method 2 was 0.77 according to evaluation "Correct". This indicates that Method 2 can extract more partly relevant answers than Method 1. When we want to extract completely relevant answers, we should use Method 1. When we want to extract more answers, including partly relevant answers, we should use Method 2.
- Method 2 was the most accurate (0.77) of those used by all eight participating teams. We could detect paragraphs as answers including input terms and the key terms related to answer types based the methods discussed in Section 3.3. Our system obtained the best results because our method of detecting answers was the most effective.

Below is an example of the output of Method 1, which was judged to be "A."

Question sentence:

jusei ran shindan wa douiu baai ni okonawareru noka?

(When is amniocentesis performed on a pregnant woman?)

System output:

omoi idenbyou no kodono ga umareru no wo fusegu.

(To prevent the birth of children with serious genetic disorders)

Examples of answers given by organizers:

omoi idenbyou

(A serious genetic disorder)

omoi idenbyou no kodomo ga umareru kanousei ga takai baai

(To prevent the birth of children with serious genetic disorders.)

5 Conclusion

We constructed a system for answering non-factoid Japanese questions. An example of a non-factoid question is "Why are people opposed to the Private Information Protection Law?" We used various methods of passage retrieval for the system. We extracted paragraphs based on terms from an input question and output them as the preferred answers. We classified the non-factoid questions into six categories. We used a particular method for each category. For example, we increased the scores of paragraphs including the word "reason" for questions including the word "why." We participated at NTCIR-6 QAC-4, where our system obtained the most correct answers out of all the eight participating teams. The rate of accuracy was 0.77, which indicates that our methods were effective.

We would like to apply our method and system to Web data in the future. We would like to construct a sophisticated system that can answer many kinds of complicated queries such as non-factoid questions based on a large amount of Web data.

Acknowledgements

We are grateful to all the organizers of NTCIR-6 who gave us the chance to participate in their contest to evaluate and improve our question-answering system. We greatly appreciate the kindness of all those who helped us.

References

Yoshiaki Asada. 2006. Processing of definition type questions in a question answering system. *Master's thesis, Yokohama National University*. (in Japanese).

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR-2000)*, pages 192–199.

Sasha Blair-Goldensohn, Kathleen R. McKeown, and Andrew Hazen Schlaikjer. 2003. A hybrid approach for qa track definitional questions. In *Proceedings of the 12th Text Retrieval Conference (TREC-2003)*, pages 185–192.

Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Susan Dumis, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Kyoung-Soo Han, Young-In Song, Sang-Bum Kim, and Hae-Chang Rim. 2005. Phrase-based definitional question answering using definition terminology. In *Lecture Notes in Computer Science 3689*, pages 246–259.

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi. 2001. IBM's Statistical Question Answering System. In *TREC-9 Proceedings*.

Julian Kupiec. 1993. MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Hideyuki Maehara, Jun'ichi Fukumoto, and Noriko Kando. 2006. A BE-based automated evaluation for question-answering system. *IEICE-WGNLC2005-109*, pages 19–24. (in Japanese).

Bernardo Magnini, Matto Negri, Roberto Prevete, and Hristo Tanev. 2002. Is it the right answer? Exploiting web redundancy for answer validation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, and Masayuki Asahara. 1999. Japanese morphological analysis system ChaSen version 2.0 manual 2nd edition.

Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154.

Kokoro Morooka and Jun'ichi Fukumoto. 2006. Answer extraction method for why-type question answering system. *IEICE-WGNLC2005-107*, pages 7–12. (in Japanese).

Masaki Murata, Kiyotaka Uchimoto, Hiromi Ozaku, Qing Ma, Masao Utiyama, and Hitoshi Isahara. 2000. Japanese probabilistic information retrieval using location and category information. *The Fifth International Workshop on Information Retrieval with Asian Languages*, pages 81–88.

Masaki Murata, Masao Utiyama, Qing Ma, Hiromi Ozaku, and Hitoshi Isahara. 2001. CRL at NTCIR2. *Proceedings of the Second NTCIR Workshop Meeting on Evaluation of Chinese & Japanese Text Retrieval and Text Summarization*, pages 5–21–5–31.

Masaki Murata, Qing Ma, and Hitoshi Isahara. 2002. High performance information retrieval using many characteristics and many techniques. *Proceedings of the Third NTCIR Workshop (CLIR)*.

National Institute of Informatics. 2002. *Proceedings of the Third NTCIR Workshop (QAC)*.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *TREC-3*.

Radu Soricut and Eric Brill. 2004. Automatic question answering: Beyond the factoid. In *In Proceedings of the Human Language Technology and Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-2004)*, pages 57–64.

TREC-10 committee. 2001. The tenth text retrieval conference. http://trec.nist.gov/pubs/trec10/t10_proceedings.html.

Jinxi Xu, Ana Licuanan, and Ralph Weischedel. 2003. TREC 2003 QA at BBN: answering definitional questions. In *Proceedings of the 12th Text Retrieval Conference (TREC-2003)*, pages 98–106.

A Multi-Document Multi-Lingual Automatic Summarization System

Mohamad Ali Honarpisheh, Gholamreza Ghassem-Sani, Ghassem Mirroshandel

Sharif University of Technology,

Department of Computer Engineering, Tehran, Iran,
Honarpisheh@ce.sharif.edu, Sani@sharif.ir, Mirroshandel@ce.sharif.edu

Abstract

Abstract. In this paper, a new multi-document multi-lingual text summarization technique, based on singular value decomposition and hierarchical clustering, is proposed. The proposed approach relies on only two resources for any language: a word segmentation system and a dictionary of words along with their document frequencies. The summarizer initially takes a collection of related documents, and transforms them into a matrix; it then applies singular value decomposition to the resulted matrix. After using a binary hierarchical clustering algorithm, the most important sentences of the most important clusters form the summary. The appropriate place of each chosen sentence is determined by a novel technique. The system has been successfully tested on summarizing several Persian document collections.

1 Introduction

With the advent of the Internet, different newspapers and news agencies regularly upload their news in their sites. This let users to access different viewpoints and quotes about a single event. At the same time of this explosive growth of the amount of textual information, the need of people for quick access to information has dramatically increased. The solution proposed for dealing with this huge amount of information is using Text Summarizers. Several systems have been developed with respect to this solution (McKeown et. al., 2002; Radev et. al., 2001).

Generally in the process of multi-document text summarization, a collection of input documents about a particular subject is received from the user and a coherent summary without redundant information is generated. However, several challenges exist in this process the most important of which are removing redundant information from the input sentences and ordering them properly in the output summary. In a new approach to multi-document summarization proposed in this paper, Singular Value Decomposition (SVD) is used to find the most important dimensions and also to remove noisy ones. This process makes clustering of similar sentences easer. In order to determine the level of importance of different clusters, the generated singular values and singular vector of the SVD have been used in a fashion similar to that (Steinberger and., Ježek, 2004). To evaluate generated summaries the SVD-based method proposed in the same paper is used.

2 Text Summarization approaches

There are different features with which we can classify text summarization systems. In (Sparck-Jones, 1999) these features are divided according to the input, purpose, and output of system. With respect to this categorization, our proposed system is a general multi-document multi-lingual text summarizer which generates extracts a summary from the input documents.

Different approaches to text summarization can be categorized in different ways according to various features of text summarization systems. With

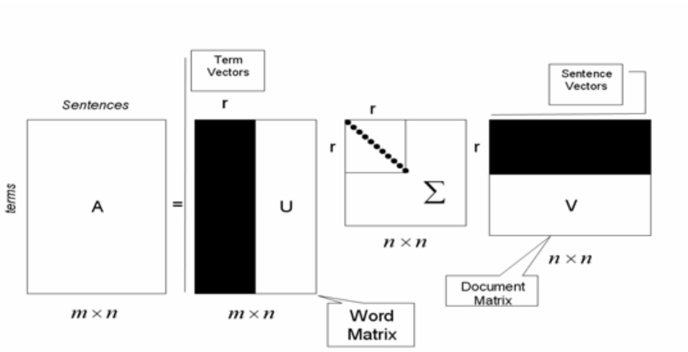


Fig. 1. Singular Value Decomposition

respect to the output of the system, there are two categories of extracting and abstracting methods. Extraction-based summarization methods are also divided into three classes.

The first group of Extraction-based methods is statistical. These methods statistically assign significance score to different textual units. The very first developed summarization methods were of this category (Edmundson and Wyllys, 1961). Scoring policy in these systems was based on different features, such as term frequency and place of the sentences. Vector Space Models (Salton et. al., 1994), compression of sentences with Automatic Translation approaches (Knight and Marcu, 2000), Hidden Markov Model (Jing and McKeown, 2000), Topic Signatures based methods (Lin and Hovy, 2000, Lacatusu et al., 2006) are among the most popular techniques that have been used in the summarization systems of this category.

The second groups of extraction-based methods, shallow understanding approaches use some information about words or textual units and their dependencies to improve the performance of extraction. The understanding can be induced using dependencies between words (Barzilay and Elhadad, 1997), rhetorical relations (Paice and Johns, 1993), events (Filatova and Hatzivassiloglou, 2004). In all of these methods, the most focused dependencies are used as a measure for saliency of each textual unit.

The third group, knowledge-based approaches, uses particular domain knowledge in discriminating the important parts of input documents. This knowledge is usually taking some assumptions about the working domain. Centrifuger (Elhadad et al., 2005) is a good example of systems in this category, which operates in medical domains.

The new approach proposed in this paper uses SVD and hierarchical clustering methods. It can therefore be categorized in the statistical based methods.

3 SVD based methods

Methods that use SVD to find salient information in the input document are a member of Vector Space Models. In such models, each textual unit is represented by a vector. Each component of this vector is filled with a value which represents both the local and global importance of each word.

The idea of using SVD in summarization was first introduced in (Gong and Liu, 2001). In this model, the input document is transformed into an $m \times n$ sparse matrix, *A*, where *m* is the number of words and *n* is the number of the sentences of input document.

The SVD of this $m \times n$ matrix, with the assumption $m \succ n$, is defined as follows:

$$A = U \Sigma V , \qquad (\mathbf{1})$$

where $U=[u_{ij}]$ is an $m \times n$ column-orthogonal matrix with columns named as the left singular vectors, $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_n)$ is an $n \times n$ diagonal matrix with non-negative diagonal elements in descending order, and $V=[v_{ij}]$ is an $n \times n$ row-orthogonal matrix with rows named as the right singular vectors (figure 1 demonstrate application of SVD to *A*). The number of non-zero elements in Σ is equal to the rank, *r*, of matrix *A*.

There are two viewpoints about the result of performing SVD on sentence by the word matrix of document (Gong and Liu, 2001). From transformation viewpoint, SVD for each sentence reduces the dimensions from *m* to *r*. The salience degree of the reduced dimension decreases from the first to the *r*th dimension. From the semantic viewpoint, SVD derives the hidden latent structure of the input document. This structure is represented in *r* linearly independent base vectors (i.e. concepts). SVD can capture and model the interrelations between concepts and hidden topics, which can be used to cluster semantically related words and sentences.

In SVD-summarization method, for each salient concept (singular vector) of matrix V, the sentence with the highest value in that dimension is chosen. This technique helps us to choose the sentences of the summary that best represent the most important concepts. For example, the most important sentence of a document in this summarization method is the sentence that has the highest value in the first row of *V*.

However, this method faces two significant problems. First, the number of dimensions should be less than or at most equal to the number of discovered. Second, in this method just individual concept is used to determine their saliency, not their combinations. This strategy obviously works poor when a sentence that is not the most important of any dimension alone may contain concept that in combination make it important enough.

These problems led to the introduction of a new summarization method (Steinberger and Ježek, 2004). This new approach uses summation of the weighted components of singular vectors instead of each individual concept alone. The weight of each vector's component of each sentence is its corresponding singular value. The reason for such weighting is to increase the effect of more important singular vectors. Formally, the degree of salience of each sentence can be computed by using the following formula:

$$s_k = \sqrt{\sum_{i=1}^{r} v_{k,i}^2 . \sigma_i^2 .} \qquad (2)$$

where s_k is the salience degree of *k*th sentence in the modified latent vector space, and *r* is the number of important dimensions of the new space. Corresponding value of each *r* dimensions is greater than half of the first singular value.

Both of above strategies for text summarization were proposed for single document summarization only. These approaches do not utilize the clustering power of SVD in discovering sentences with close meanings. On the other hand, their pure reliance on SVD, which does not depend on the characteristics of any language, makes them appropriate to be applied to any language.

4 Multi-Document SVD based Summarization

In this paper a new version of the SVD based summarization is introduced. After transforming all documents into sentence by word matrix, SVD is applied to the resultant matrix. To remove redundant sentences from the summary, a hierarchical bottom-up clustering algorithm is applied to the *r* most important extracted concepts of the input sentences. After extracting the clusters, the saliency score of each cluster is determined, and the most important clusters are selected. At the same time, using a simple ordering method, the appropriate place of each sentence in the sorted collection is determined. In the following sections, each of these processes is described in more details.

Matrix Construction and Application of SVD

Given a collection of input documents that are related to a particular subject, the system decomposes the documents into sentences and their corresponding words. In addition, it computes the total number of occurrences of each word in all documents as the Term Frequency (TF).

The developed system works on Persian language. It is assumed that words are separated by spaces. However, some words in Persian are compound words .However this did not cause any problem for the developed system; because the most meaningful part of such words is usually less common than others and thus have an Inverse Document Frequency (IDF) that is higher than that of more common less meaningful parts. IDF represents the amount of meaning stored in each word. It can be used to reduce the impact of less important constituents such as stop words, which usually have a high TF but contains little meaning.The formula for calculating IDF is as follows:

$$IDF(term) = \log(\frac{NUMDOC}{NUMDOC(term)}), \qquad (3)$$

where *NUMDOC* represents the total number of document processed to create IDF dictionary and *NUMDOC(term)* is the number of documents in which the *term* appeared.

After decomposition, the input sentences along with their corresponding words are arranged as a matrix. Two different weighting schemes have been applied to each element of this matrix: 1) a constant value and, 2) each word's associated TF*IDF (Salton and Buckley, 1988).

After constructing the Sentence by word matrix, SVD is applied to the resultant matrix. Applying SVD removes the effect of unimportant words and highlights more salient dimensions. It also reduces the number of dimensions for each sentence, resulting in an easer clustering process. This improves the performance of sentence clustering by making it faster and less sensible to unimportant differences between sentences.

Clustering

To cluster reduced dimension sentences, a binary hierarchical agglomerative clustering algorithm with average group linkage is used. In this algorithm, at first, each sentence is considered as a cluster. At each step, two closest clusters are combined into a single cluster. The dimension of this new cluster is the average dimensions of the two combining ones. These steps are repeated until we have only one cluster. So the result of this algorithm is a binary tree.

The question that needs to be answered at this step is "how clusters containing similar sentences can be extracted from this binary tree?" Two properties are required to propose a sub-tree as a possible cluster of similar sentences:

1. The number of existing sentences at the current node (cluster) should be less than or equal to the total number of input documents; because it is assumed that there is not much redundant information in each document. This assumption is valid with respect to the news documents in which there might be only little redundancy.
2. The distance between two children of the current cluster should be less than or equal to the distance between current cluster and its sibling node. This condition has been found empirically.

Using these two heuristics, similar clusters are extracted from the binary tree.

Finding Salient units

To select important clusters from the set of extracted clusters, different clusters are scored based on two diffrent methods. In the first method, the average of TF*IDF of different words in the each sentence in the current cluster are used. The second approach was the latest SVD-based approach which was proposed by (Steinberger and Ježek, 2004) and was described in the section 3. In the latter, score of each cluster is found using the following formula:

$$score(cluster) = \frac{\sum_{s \in cluster} score(s)}{|cluster|}, \qquad \textbf{(4)}$$

where $|cluster|$ represent the number of sentences in the current cluster.

Selecting and ordering sentences:

In this step, the final representation of the summary will be generated. To this end, from each important cluster, a sentence should be selected. At the same time, the proper order of selected sentences should be determined. To find this order, a Representative Document (RD) is selected. The RD is the document which includes most sentences of the most important clusters. After selecting RD the following steps are performed:

1. Starting from the most important cluster, while the number of processed words of summary sentences does not exceed form the specified number:
   a. If no sentence from the current cluster was not added to the summary:
      i. If there is a sentence from the RD in this cluster, choose this sentence;
      ii. Otherwise find the most important sentence, the current cluster: To find out the place of the selected sentence in the summary, a search is performed for clusters that contain both sentences from RD and neighbors of the selected sentence. The place of the sentences from RD that co- clustered with neighbors of the selected sentence is chosen as the selected sentence boundary.
      iii. If any place has been found for the selected sentence, add it to summary in the specified location, and mark that cluster as having a sentence in the summary.

2. If it remains any unplaced sentence which should be presented in the summary, go to step 1 with the remaining number of words.

5 Experiments

5.1 Testing Collection

The proposed summarizer is originally developed for the Persian language. In Persian like many other languages there is not a standard test collection, to evaluate the summarizers. To overcome the lake of a test collection in Persian, an unsupervised approach of evaluating summaries is selected (i.e. SVD-based evaluation method proposed in (Steinberger and Ježek, 2004)). In addition to an evaluator, a collection of documents was also required. For this purpose different texts related to a single event were collected. The properties of these collections are presented in table 1

5.2 Results and Discussions:

To find out which term weighting and distance measure causes the highest increase in the SVD-Scores, various combinations of these approaches has been used in the summarization approach. To find the distance between clusters, Euclidian, Hamming, and Chebyshev distances and to determine the saliency of different clusters, TFIDF and SVD-based methods were used. The gained SVD-Based score using different configurations are represented in table 2

As it can be seen in table 2, TFIDF-based methods score higher than SVD-based methods. Also, the most promising distance was the hamming distance. It can also be seen that the performance decreases substantially when instead of a constant value *tf*idf* scores were used. It was observed that using various distance measure the SVD-Score of each collection would be different. The SVD-Scores are in favor of using the boosting methods for classification of sentences with different distance measure for each classifier. Comparing these results with the ones proposed in (Steinberger and Ježek, 2004), a significant decrease in evaluated SVD-Based scores is observed. One of the reasons for this phenomenon is that the distinct words appearing in multi-documents are more extensive than the words appear in a single document.

6 Conclusions

This paper presents a new SVD-based multi-lingual multi-document summarization method using agglomerative clustering. In this method, first, using SVD, the most important concepts representing sentences are extracted into a word by a sentence matrix. Then, similar sentences are clustered using a new heuristic method. Finally, impor-

Average number of distinct words in documents	1474
Average number of sentences in documents	32
Average number of sentences in subjects	643
Maximum Number of distinct words in subjects	2189
Minimum Number of distinct words in subjects	485
Number of subjects	14

Table 1. Testing Collections –Details

	Euclidian			Hamming			Chebyshev		
	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min
TFIDF	0.450	0.572	0.286	**0.518**	0.596	0.343	0.475	0.605	0.264
SVD-based	0.466	0.632	0.313	0.472	0.650	0.322	0.449	0.620	0.309

Table 2. Using a constant value for word-sentence matrix

	Euclidian			Hamming			Chebyshev		
	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min
TFIDF	0.364	0.549	0.109	0.269	0.512	0.113	0.406	0.512	0.283
SVD-based	0.309	0.134	0.563	0.319	0.499	0.112	0.367	0.518	0.235

Table 3. Using TF-IDF for each element of the matrix

in the summary are extracted from the resulting clusters. Different weighting schemes, distance metrics and scoring methods have been experimented. According to our experiments constant weighting scheme along with hamming distance is superior to other combinations. Since this method only needs determination of words and their inverse document frequency, it can be applied to any language providing these resources. We are now trying to improve the performance of the proposed algorithm. It seems that applying Principle Direction Partitioning (Blei, 2002) algorithm in the clustering phase and using Latent Dirichlet Allocation method (Boley, 1998) instead of the SVD based ones to model sentences can improve the score of the proposed method.

References

Barzilay, R. and Elhadad. M. 1997. “Using lexical chains for text summarization.” In Proceedings of the ACL/EACL'97 Summarization Workshop, Madrid, Spain.

Blei, D., Ng, A., and Jordan M., Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993–1022, January 2003. (A shorter version appeared in NIPS 2002).

Boley, D.L.: Principal Direction Divisive Partitioning. Data Mining and Knowledge Discovery, Vol. 2(4):325–344, Dec. 1998.

Edmundson, H.P. and Wyllys, R.E., Automatic abstracting and indexing - Survey and recommendations, Communications of the ACM, Vol. 4, (1961) 226-234

Elhadad, N., Kan, M.Y., Klavans, J., McKeown, K.: Customization in a unified framework for summarizing medical literature, Artificial Intelligence in Medicine Vol. 33(2): (2005) 179-198.

Filatova, E., Hatzivassiloglou, V.: Event-based Extractive summarization, In: Proceedings of ACL 2004 Workshop on Summarization, (2004) 104-111.

Gong, Y., Liu, X.: Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. Proceedings of the 24th ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States (2001) 19-25

Jing, H., McKeown, K.: Cut and paste based text summarization. Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'00), (2000), Seattle-Washington

Knight, K., Marcu, D.: Statistics based summarization .step one: Sentence compression, Proceeding of the 17th National Conference of the American Association for Artificial Intelligence (2000)703-710.

Lacatusu, F., Hickl, A., Roberts, K., Shi, Y., Bensley, J., Rink, B., Wang, P., Taylor, L.: LCC’s GISTexter at DUC 2006: Multi-Strategy Multi-Document Summarization, Document Understanding Conference (2006)

Lin, C.Y., Hovy, E.: From single to multi-document summarization: A prototype system and its evaluation. Proceedings of the ACL, pages 457–464, 2002

McKeown, K.R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Klavans, J.L., Nenkova, A., Sable, C., Schiffman, B., Sigelman, S.: Tracking and summarizing news on a daily basis with Columbia's newsblaster. Proceedings of 2002 Human Language Technology Conference (HLT), San Diego, CA, 2002

Paice, C. D., Johns, A. P.: The identification of important concepts in highly structured technical papers, In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (1993)

Radev, D. R., Blair-Goldensohn, S., Zhang, Z., Raghavan R.S.: Newsinessence: A system for domain-independent, real-time news clustering and multi-document summarization. Proceedings of 2001 Human Language Technology Conference (Demo Session), San Diego, CA, 2001

Salton, G., Allan, J., Buckley, C., Singhal, A.: Automatic analysis, theme generation, and summarization of machine readable texts, Science, Vol. 264(5164), (1994) 1421–1426

Salton, G. and Buckley, C.. Term weighting approaches in automatic text retrieval. Information Processing and Management, (1988), 24(5):513523

Sparck-Jones, K.: Automatic summarizing: factors and directions. Mani I, Maybury MT, editors. Advances in automatic text summarization. (1999) 10-12 [chapter 1]

Steinberger, J., Ježek, K. : Text Summarization and Singular Value Decomposition, Lecture Notes in Computer Science.Advances in Information Systems, Vol. 3261/2004, Springer-Verlag, Berlin Heidelberg New York (2004) 245-254

Summarization by Analogy: An Example-based Approach for News Articles

Megumi Makino and **Kazuhide Yamamoto**
Dept. of Electrical Engineering, Nagaoka University of Technology
1603-1 Kamitomioka, Nagaoka, Niigata 940-2188 Japan
{makino,ykaz}@nlp.nagaokaut.ac.jp

Abstract

Automatic summarization is an important task as a form of human support technology. We propose in this paper a new summarization method that is based on example-based approach. Using example-based approach for the summarization task has the following three advantages: high modularity, absence of the necessity to score importance for each word, and high applicability to local context. Experimental results have proven that the summarization system attains approximately 60% accuracy by human judgment.

1 Introduction

The example-based approach generates language by *imitating* instances, which originated in the machine translation method based on the analogy (Nagao, 1984). The idea is derived from the observation that a human being translates according to past translation experiences. In the machine translation task, this approach has been implemented, and has so far achieved efficient results (Sumita, 1998; Imamura, 2004).

In summarization, a human being also summarizes with his own knowledge and experiences. For this reason, we focus on a summarization method which is based on analogy, *example-based summarization*. The example-based method summarizes the input text in three steps. First, it retrieves a similar instance to the input text. Second, it links equivalent phrases between the input text and the similar instance. Finally, a summary is acquired with combination of some corresponding phrases. Here, we employed a Japanese news article as the input text and utilized news headlines as the instances. The news headline consists of one brief sentence which describes the main point.

We assert that the example-based summarization has the following advantages:

(1)High modularity

Easy improvement and maintenance are required to formulate a useful system in general. An example-based framework makes it easy for us to improve a system by only adding instances. Besides, the addition of instances causes few side-effects.

(2)Use of similarity rather than importance

Almost all previous work on summarization has focused on a sentence extraction. These works compute importance for each word to extract a sentence. However, it is difficult to compute the importance which correlates with human sense. Example-based summarization means there is no need to measure the importance, and it computes the similarity instead. We think it is easier to assess the similarity between two expressions rather than the importance of one expression.

(3)High applicability to local context

The statistical method, in general, attempts to compute the probability of each word appearing in the summary corpus (Knight and Marcu, 2002; Witbrock and Mittal, 1999). This may increase difficulties in maintaining local context, since the statistical approach focuses on the global probability. However, the example-based approach attempts to find most locally similar instance out of the instance collection, which may increase the fitness of input contexts.

For the three reasons given above, this paper explains the system which summarizes a Japanese news article to a one-sentence summary by imitating the similar instance.

As related work, Nguyen et al. (2004) have proposed an example-based sentence reduction model. They deal with the compression of one sentence, while we summarize some sentences into a one-sentence summary. Thus, our summarization ratio is inevitably lower than theirs, as it is considered to be more difficult as a summarization task.

Many studies have summarized some sentences, such as a news article, into a one-sentence summary. Most of them extract the important sentence and contract it. In contrast, our method generates a one-sentence summary by combining phrases in some sentences. Consequently, we can obtain high compression summaries that include information from many positions of the source.

2 Instance Collection

Our example-based summarization regards news headlines as the instance collection. A news headline is a short sentence in which the primary point is written. The following example is Japanese news headlines:

Example (1) :
三菱自動車工業は、中国で乗用車を生産へ。
(Mitsubishi Motors Corp. produces passenger cars in China.)

We use Japanese news headlines, like the above examples, as instances. Besides, as we have noted, only news headlines are used as instances; that is, the pairs formed by an original sentence and its summarized sentence are not used.

3 Example-based Summarization

3.1 Overview

Our example-based summarization system summarizes a lengthy news article into a one-sentence summary by using instances. The overall process is illustrated in figure 1. The system is composed of the following three processes in this order:

1. Retrieve a similar instance to an input news article from the instance collection.
2. Align corresponding phrases between the input news article and the similar instance.
3. Combine the corresponding phrases to form a summary.

Detail of each process is described hereafter.

3.2 Retrieval of Similar Instance

The system measures a similarity between the input and each instance in the instance collection when it retrieves a similar instance. If many words are shared between two expressions, we regard two expressions as similar. Hence, the similarity is calculated on basis of the overlaps of content words between the input news article I and the instance E, defined as follows:

$$Sim(E,I)=\sum_{i=1}^{n} Score(i)\cdot\{w\cdot||Tv_1(E)\cap Tv_i(I)|| + ||To_1(E)\cap To_i(I)||\} \quad (1)$$

where,
- n : the number of sentences in input,
- $Tv_i(\cdot)$: the verbs set in the last phrase of the i-th sentence,
- $To_i(\cdot)$: the set of content words in the i-th sentence,
- $||Tv_1(E)\cap Tv_i(I)||$: the number of overlaps between $Tv_1(E)$ and $Tv_i(I)$.

In the equation, $Score(i)$ and w are designed to give a higher score if words indicating the main topic of the input article are matched with words in the instance. We have found that words have different contributions, depending on the sentence position, to the main topic. Therefore, we apply $Score(i)$ which depends on the sentence position i, and we use the following experimentally-determined score as $Score(i)$.

$$Score(i)=\begin{cases} 5.15 & \text{if} \quad i=1 \\ 2.78/i^{0.28} & \text{otherwise} \end{cases} \quad (2)$$

The score indicates an agreement rate of content words depending on the sentence position, which is calculated by using 5000 pairs of newspaper's body and its title[1] We have also found that the verbs in the last phrase are appropriate for the main topic of the input article. For that reason, we determine the weight $w=3$ by our experiment.

Example 2 shows the similar instance obtained by measuring the similarity.

Example (2) :
Input news article
品質管理能力などの再強化策を話し合うものづくり懇談会が24日、会合を開催した。(skip the

[1] We used the same kind of newspaper as data set in section 4.1 for calculating $Score(i)$.

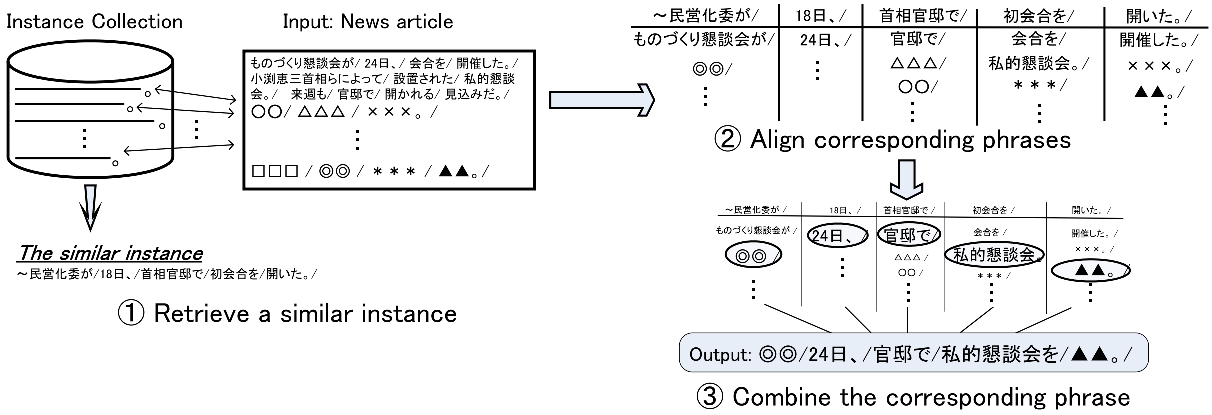


Figure 1: Overview of example-based summarization

rest.)
(The Manufacturing Council held a meeting on the 24th, which discusses the hard-hitting strategy for quality management. ...)

Obtained similar instance
道路公団民営化委が 18 日、首相官邸で初会合を開いた。
(The committee for the privatization of the Public Roads Administration held the first meeting on the 18th at the prime minister's office.)

3.3 Phrase Alignment

We compare the phrases in the input with those in the similar instance, and the system aligns the corresponding phrases. Here, the correspondence refers to the link of the equivalent phrases between the input and its similar instance. The detail of phrase alignment procedures are shown in the following.

To begin with, sentences both in the input and in the similar instance are analyzed using a Japanese syntactic parser CaboCha[1)]. The sentences are split into phrases and named entities (NEs), such as *PERSON*, *LOCATION*, *DATE*, are recognized by the tool.

Then the adnominal phrases in the similar instance are deleted. This is because the adnominal phrases are of many types, depending on the modified noun; accordingly, the adnominal phrase should be used only if the modified nouns are exactly matched between the input and the similar instance.

Finally, the system links the corresponding phrases. Here, phrase correspondence is one-to-many, not one-to-one, and therefore a phrase in a similar instance has some corresponding phrases in the input. In order to compare phrases, the following four measures are employed: (i) agreement of grammatical case, (ii) agreement of NE, (iii) similarity with enhanced edit distance, and (iv) similarity by means of mutual information. The measure of (i) focuses on functional words, whereas the measures of (ii)-(iv) note content words. Let us explain the measures using example 2.

(i) Agreement of Grammatical Case

If there is a phrase which has the same grammatical case[2] in the input and in the similar instance, we regard the phrase as the corresponding phrase. In example 2, for example, the phrases "再強化策 を (the hard-hitting strategy `obj`[3]), 会合 を (the meeting `obj`)" in the input corresponds the phrase "初会合 を(the first meeting `obj`)" in the similar instance.

(ii) Agreement of Named Entity

Provided the input has the same NE tag as the similar instance, the phrase involving its tag links the corresponding phrase. For example, in example 2, the phrase "24 日 [*DATE*] (on the 24th.)" in the input corresponds the phrase "18 日 [*DATE*] (on the 18th.)" in the similar instance.

(iii) Similarity with Enhanced Edit Distance

We adopt the enhanced edit distance to link phrases including the same characters, because Japanese abbreviation tends to include the same characters as the original. For example, the abbreviation of "日

[2] Comma is also regarded as grammatical case (i.e., null case) here.
[3] "`obj`" is an object case marker.

本銀行 (Bank of Japan)” is “日銀”. The enhanced edit distance is proposed by Yamamoto et al. (2003). The distance is a measure of similarity by counting matching characters between two phrases. Moreover, the distance is assigned a different similarity weight according to the type of matched characters. We apply 1.0 to the weight only if Chinese-derived characters (Kanji) are matched. We link phrases as corresponding phrases, where the phrases are the top three similar to a phrase in the similar instance.

(iv) Similarity with Mutual Information

We finally compute the similarity with mutual information to link syntactically similar phrases. For example, given the following two expressions: “会議を開く (to hold a meeting)” and “大会を開く (to hold a convention)”, we regard 会議 *(a meeting)* and 大会 *(a convention)* as similar. We use the similarity proposed by Lin (1998). The method uses mutual information and dependency relationships as the phrase features. We extend the method to Japanese by using a particle as the dependency relationships. We link phrases as corresponding phrases, where the phrases are the top three similar to a phrase in the similar instance.

3.4 Combination of the Corresponding Phrases

Our system forms the one-sentence summary by combining the corresponding phrases. Let us explain this process by using figure 2. We arrange the phrase of the input on the node, where the phrases is judged as the correspondence to the phrase in the similar instance. For example, in figure 2, the second nodes *e* and *d* denote the corresponding phrases in the input, which correspond to the second phrase *had* in the similar instance.

We assign the similarity between corresponding phrases as the weight of node. In addition to this, we employ phrase connection score to the weight of edge. The score indicates the connectivity of consecutive two phrases, e.g. two nodes such as node *d* and node *e* in figure 2. If you want to obtain a fine summary, i.e., a summary that contains similar phrases to the similar instance, and that is correct grammatically, you have to search the best path $\hat{W}_p$ for path sequence $W_p = \{w_0, w_1, w_2, \cdots, w_m\}$, where the best path maximizes the score.

$$\hat{W}_p = W_p \quad \text{s.t.} \quad \underset{p}{\operatorname{argmax}}\ Score_p(W_p) \tag{3}$$

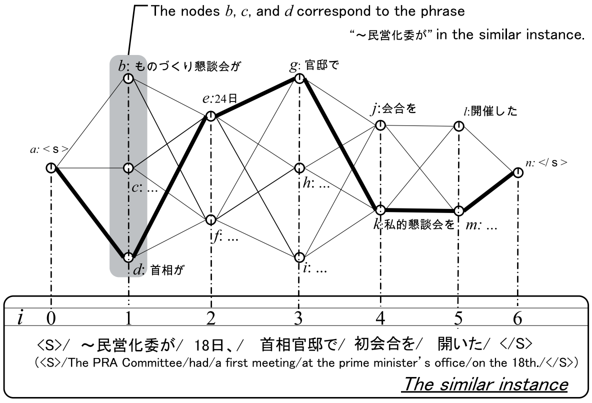


Figure 2: Optimal path problem that depends on combination of the corresponding phrases[4].

The best path $\hat{W}_p$ is a one-sentence summary which is generated by our system. Take the case of the thick line in figure 2, $\hat{W}_p$ is indicated as $\hat{W}_p = \{a, d, e, g, k, m, n\}$, namely, generated summary is formed the phrases $\{a, d, e, g, k, m, n\}$. In eq.3, $Score_p(W_p)$ is given by

$$Score_p(W_p) = \alpha \sum_{i=0}^{m} N(w_i) + (1-\alpha) \sum_{i=1}^{m} E(w_{i-1}, w_i) \tag{4}$$

where α is the balancing factor among the weights of node and edge. We score $\alpha = 0.6$ by our experiment. m indicates the last number of the phrase in the similar instance, $N(w_i)$ is given as follows:

$$N(w_i) = \max \begin{cases} 0.5 & \text{if (grammatical case or NE tag is matched)} \\ 1/rank & \text{otherwise} \end{cases} \tag{5}$$

where, *rank* indicates the rank order of the similarity with the enhanced edit distance or mutual information to the phrase w_i. $N(w_i)$ illustrates the similarity between corresponding two phrases. The node score, shown above, is determined by the preliminary experiment. The edge score $E(w_{i-1}, w_i)$ is given by

$$E(w_{i-1}, w_i) = \frac{1}{|loc(w_{i-1}) - loc(w_i)| + 1} \tag{6}$$

where, $loc(w_i)$ denotes where the location of the sentence contains the phrase w_i in the input. The edge score means that if w_{i-1} and w_i are located closely to each other, a higher score is given, since a good connection is expected in this case.

[4]The nodes, *a*, *b*, *c*, $\cdots$, *n*, indicate the corresponding phrases to the phrase in the similar sentence. For example, the nodes, *b*, *c*, *d* correspond to “*The PRA Committee.*” *i* is a phrase number in the similar sentence.

4 Evaluation and Discussion

4.1 The Corpus

We used 26,784 news headlines as instances, which were collected from the Nikkei-goo mail service[2)] for 2001-2006. In order to adjust the weight w in the eq.1 and the balancing parameter α in eq.4, 150 input news articles were used as the tuning set. A different group of 134 news articles were used for evaluation. We used Nihon Keizai Shimbun, a Japanese newspaper[3)], from 1999 through 2000 as tuning and test data.

4.2 Summarization Ratio

To calculate summarization ratio, we have compared the number of characters in the input news articles with that in the output summary. As the result, we obtained a summarization ratio of 5%; namely, 95% characters in the input were reduced. From the summarization ratio, our approach made it possible to summarize sentences into one-sentence summary with high compression.

4.3 Sectional Evaluation

We evaluated each part of our system by human judgment[5]. We first evaluated the process by retrieving similar instance. Next, we evaluated the processes of phrase alignment and the combination by assessing whether the output summaries were appropriate.

· Retrieving Process

An examinee evaluated the similar instances obtained. Given an input news article and the similar instance to the input, the examinee rates the following scale from one to four, based on how similar the similar instance obtained is to the summary which the examinee generated from the input news article:

1) quite similar 2) slightly similar
3) not very similar 4) not similar

Out of 134 input articles, 77 inputs were ranked either 1) quite similar or 2) slightly similar. As a consequence, the accuracy of similar instance obtained is approximately 57%, which indicates that the similarity calculation for obtaining similar instance is feasible.

[5] One examinee judged the parts of our system.

· Phrase Alignment and Combination

We also evaluated parts of phrase alignment and the combination by human judgment. The examinee compared 77 output summaries with their input. Here, we limited 77 outputs judged as good similar instances in evaluation of the process of retrieving similar instance, because we evaluate specifically the parts of phrase alignment and combination.

The examinee categorized them based on how proper the output summary is to the input news article:

1) quite proper 2) slightly proper
3) not very proper 4) not proper

As a result of judgment, 48 outputs out of 77 are evaluated either 1) quite proper or 2) slightly proper.

Both a statistical method by Knight and Marcu (2002) and an example-based method by Nguyen et al. (2004) contracted one-sentence with a summarization ratio of approximately 60-70%. Both papers indicated that a score of 7-8 on a scale from one to ten was obtained. They deal with the compression of one sentence, while we summarize some sentences into a one-sentence summary. Thus, our summarization ratio is lower than theirs, as it is considered to be more difficult as a summarization task. Despite this, we obtained the ratio that 62% (48 out of 77 results) were judged proper. Although direct comparison of the performance is impossible, it is considered that our proposed method obtains a competitive accuracy.

4.4 Discussions

· Examples of Output Summary

Figure 3 shows some examples of the output summary.

From figure 3, we can see that the similar instances were effectively used, and the appropriate summaries to the input are generated. For example, the second summary in the figure is judged as a fine summary contracting information of two sentences according to the similar instance.

· Analysis of Summarization Errors

In the course of our summarization, we have observed errors due to erroneous correspondences. In Japanese, sometimes two or more phrases are contracted into one phrase, as in the example below. We now only attempt to correspond two phrases one by

Input news article：
神奈川県警の一連の不祥事のうち、厚木署集団警ら隊の集団暴行事件で起訴された元巡査部長、川野優被告の論告求刑公判が二十一日、横浜地裁で開かれた。検察側はひまを持て余して部下に短銃を突き付けるなど、組織における地位の高さに乗じた悪質な行為などと理不尽な暴力を指弾し、川野被告に懲役一年六月を求刑した。(skip the rest.)
(The prosecution made Kawano's closing arguments on the 21st in the trial at the Yokohama District Court. The ex-sergeant Suguru Kawano is accused of gang-bashing by Atsugi Police Station's patrol group in a string of scandals of Kanagawa Prefectural Police. The prosecutors demanded one and half year in a prison. ...)
Obtained similar instance：
大阪地裁で 22 日、8 人が犠牲となった池田小児童殺傷事件の論告求刑公判が開かれ、検察側は宅間被告に死刑を求刑した。
(The prosecution made Takuma's closing arguments on the 22nd in the trial at the Osaka District Court, and asked for the death penalty.)
Output summary：
横浜地裁で二十一日、論告求刑公判が開かれ、検察側は川野被告に懲役一年六月を求刑した。
(The prosecution made Kawano's closing arguments on the 21st in the trial and demanded one and half years in prison.)

Figure 3: The examples of generated summary

one, and we thus can not deal with many-to-one correspondences.

Example (3) :
前年同月に/[6]比べ → 前年同月比/
(compare with the same month last year)
5 月を/メドに → 5 月メド/
(in May)

We expect that this kind of phenomenon can be solved by paraphrasing an input summary as well as summary instance. Recently, several works on paraphrasing techniques have been proposed in Japanese, hence such pre-processing before alignment would be feasible.

5 Conclusion and Future Work

We have presented an example-based technique that has been applied to the summarization task. The essence of the proposed method is to generate a one-sentence summary by combining instances each of which imitates the given input.

As the result of human judgment, the retrieval process of a similarity sentence attained 57% accuracy. And our method generated summary in which 62% were judged proper. We have confirmed by our observation that the summaries were generated by combining the phrases in many positions of the input, while those summaries are not given just by common methods such as sentence extraction methods and sentence compression methods.

The sectional evaluation and the inspection of example output show that this system works well. However, larger scale evaluation and comparison of its accuracy remain to be future work.

[6]"/" indicates a phrase boundary.

Tools and language resources

1) CaboCha, Ver.0.53, Matsumoto Lab., Nara Institute of Science and Technology.
http://chasen.org/˜taku/software/cabocha/
2) Nikkei News Mail, NIKKEI-goo,
http://nikkeimail.goo.ne.jp/
3) Nihon Keizai Shimbun Newspaper Corpus, years 1999–2000, Nihon Keizai Shimbun, Inc.

References

Kenji Imamura. 2004. *Automatic Construction of Translation Knowledge for Corpus-based Machine Translation*. Ph.D. thesis, Nara Institute of Science and Technology.

Kevin Knight and Daniel Marcu. 2002. Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. *Artificial Intelligence*, 139(1):91–107.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of COLING-ACL98*, pages 768–773.

Makoto Nagao. 1984. A Framework of a Mechanical Translation Between Japanese and English By Analogy Principle. In *Artificial and Human Intelligence*, pages 173–180.

Minh Le Nguyen, Susumu Horiguchi, Akira Shimazu, and Bao Tu Ho. 2004. Example-Based Sentence Reduction Using the Hidden Markov Model. *ACM Transactions on Asian Language Information Processing*, 3(2):146–158.

Eiichiro Sumita. 1998. *An Example-Based Approach to Transfer and Structural Disambiguation within Machine Translation*. Ph.D. thesis, Kyoto University.

Michael J. Witbrock and Vibhu O. Mittal. 1999. Ultra-Summarization: A Statistical Approach to Generating Highly Condensed Non-Extractive Summaries. In *Research and Development in Information Retrieval*, pages 315–316.

Eiko Yamamoto, Masahiro Kishida, Yoshinori Takenami, Yoshiyuki Takeda, and Kyoji Umemura. 2003. Dynamic Programming Matching for Large Scale Information Retrieval. In *Proceedings of the 6th International Workshop on Information Retrieval with Asian Languages*, pages 100–108.

Sentence Ordering based on Cluster Adjacency in Multi-Document Summarization

Ji Donghong, Nie Yu
Institute for Infocomm Research
Singapore, 119613
{dhji, ynie}@i2r.a-star.edu.sg

ABSTRACT

In this paper, we propose a cluster-adjacency based method to order sentences for multi-document summarization tasks. Given a group of sentences to be organized into a summary, each sentence was mapped to a theme in source documents by a semi-supervised classification method, and adjacency of pairs of sentences is learned from source documents based on adjacency of clusters they belong to. Then the ordering of the summary sentences can be derived with the first sentence determined. Experiments and evaluations on DUC04 data show that this method gets better performance than other existing sentence ordering methods.

1. Introduction

The issue of how to extract information from source documents is one main topic of summarization area. Being the last step of multi-document summarization tasks, sentence ordering attracts less attention up to now. But since a good summary should be fluent and readable to human being, sentence ordering which organizes texts into the final summary could not be ignored.

Sentence ordering is much harder for multi-document summarization than for single-document summarization (McKeown et al., 2001; Barzilay and Lapata, 2005). The main reason is that unlike single document, multi-documents don't provide a natural order of texts to be the basis of sentence ordering judgment. This is more obvious for sentence extraction based summarization systems.

Majority ordering is one way of sentence ordering (McKeown et al., 2001; Barzilay et al., 2002). This method groups sentences in source documents into different themes or topics based on summary sentences to be ordered, and the order of summary sentences is determined based on the order of themes. The idea of this method is reasonable since the summary of multi-documents usually covers several topics in source documents to achieve representative, and the theme ordering can suggest sentence ordering somehow. However, there are two challenges for this method. One is how to cluster sentences into topics, and the other is how to order sentences belonging to the same topic. Barzilay et al. (2002) combined topic relatedness and chronological ordering together to order sentences. Besides chronological ordering, sentences were also grouped into different themes and ordered by the order of themes learned from source documents. The results show that topic relatedness can help chronological ordering to improve the performance.

Probabilistic model was also used to order sentences. Lapata (2003) ordered sentences based on conditional probabilities of sentence pairs. The conditional probabilities of sentence pairs were learned from a training corpus. With conditional probability of each sentence pairs, the approximate optimal global ordering was achieved with a simple greedy algorithm. The conditional probability of a pair of sentences was calculated by conditional probability of feature pairs occurring in the two sentences. The experiment results show that it gets significant improvement compared with randomly sentence ranking.

Bollegala et al. (2005) combined chronological ordering, probabilistic ordering and topic relatedness ordering together. They used a machine learning approach to learn the way of combination of the three ordering methods. The combined system got better results than any of the three individual methods.

Nie et al. (2006) used adjacency of sentence pairs to order sentences. Instead of the probability of a sentence sequence used in probabilistic model, the adjacency model used adjacency value of sentence pairs to order sentences. Sentence adjacency is calculated based on adjacency of feature pairs within the sentence pairs. Adjacency between two sentences means how closely they should be put together in a set of summary sentences. Although there is no ordering information provided by sentence adjacency, an optimal ordering of summary sentences can be derived by use of adjacency information of all sentence pairs if the first sentence is properly selected.

In this paper, we propose a new sentence ordering method named cluster-adjacency based ordering. Like the feature-adjacency based ordering mentioned above, the ordering process still depends on sentence adjacency. But we cluster sentences first and use cluster adjacency instead of feature adjacency to calculate sentence adjacency. The advantage of this change is to avoid the sensitivity of the adjacency

information to limited number of individual features, which usually needs manual intervention.

The remainder of this paper is organized as follows. In section 2, we specify the motivation of this method. In section 3, we talk about the sentence classification using a semi-supervised method. In section 4, we discuss the procedure for sentence ordering. In section 5, we present experiments and evaluation. In section 6, we give the conclusion and future work.

2. Motivation

Majority ordering assumes that sentences in the summary belong to different themes or topics, and the ordering of sentences in the summary can be determined by the occurring sequence of themes in source documents. In order to derive the order of themes, Barzilay et al. (2002) presented themes and their relations as a directed graph. In the graph, nodes denote themes; an edge from one node to another denotes the occurring of one theme before another in a source document, and the weight of an edge is set to be the frequency of the theme pair co-occurring in the texts. Each theme is given a weight that equals to the difference between its outgoing edges and incoming edges. By finding and removing a theme with the biggest weight in the graph recursively, an ordering of themes is determined.

Probabilistic ordering method treats the ordering as a task of finding the sentence sequence with the biggest probability (Lapata, 2003). For a sentence sequence T= $S_1, S_2, \ldots, S_n$, suppose that the probability of any given sentence is determined only by its previous sentence, the probability of a sentence sequence can be generated based on the condition probabilities $P(S_i|S_{i-1})$ of all adjacent sentence pairs in the sequence. The condition probability $P(S_i|S_{i-1})$ can be further resolved as the product of condition probabilities of feature pairs $P(f_l|f_m)$, where f_l is the feature in S_i, f_m is the feature in S_{i-1}.

By finding the sentence with the biggest condition probability with the previous one recursively, an ordering of sentences is determined. A null sentence is normally introduced at the beginning of each source document to find the first sentence (Lapata, 2003).

Both majority ordering and probabilistic ordering determine text sequences in the summary based on those in the source documents. The intuition behind the idea is that the ordering of summary sentences tends to be consistent with those of document sentences. However, we notice that some important information might be lost in the process. Consider examples below:

Example 1: *Source Document = ……ABA……*

Example 2: *Source Document 1 = ……AB……*

Source Document 2 = ……BA……

Here *A* and *B* denote two themes. Let's assume that *A* and *B* are both denoted by the summary sentences. In both examples, the frequency of *A* preceding *B* equals to that of *B* preceding *A*, thus no sequence preference could be learned from the two examples, and we can only estimate a probability of 0.5 following one by another. With such estimation, the intuition that *A* and *B* shall be put adjacently although their ordering is not clear would be difficult to capture.

An adjacency based ordering (Nie et al., 2006) was proposed to capture such adjacency information between texts during sentence ordering. It uses adjacency of sentence pairs to order summary sentences. Adjacency between two sentences can be seen as how closely they should be put together in an output summary. In general, sentence adjacency is derived from that of feature pairs within sentences. Note that there is no clue to decide the sequence of two sentences purely based on their adjacency value. However, if the first sentence has been decided, the total sentence sequence can be derived according to the adjacency values by recursively selecting one having the biggest adjacency value with the most recently selected.

For adjacency based ordering, a problem is how to calculate the adjacency value between two sentences. For feature-adjacency based ordering, the sentence adjacency is calculated based on that of feature pairs within the two sentences. But a sentence may contain many single word features, and there may exist many noisy features, especially for longer sentences. To eliminate the impact of noisy features, one simple method is to select top *n* most adjacent feature pairs among the two sentences (Nie et al., 2006). However, the parameter heavily influences the performance, as shown in Table 1, where each row gives the result of a run with the same window range and different top *n* adjacent feature pairs.

Win_ range	τ(top-n=1)	τ(top-n=2)	τ(top-n=3)	τ(top-n=4)	τ(top-n=5)	τ(top-n=10)
2	**0.184**	0.213	0.253	**0.262**	0.261	0.224
3	0.251	0.252	0.273	0.274	0.257	0.213
4	**0.201**	0.253	0.268	**0.316**	0.272	0.248

Table 1. Feature-Adjacency Based Ordering

The heavy reliance on the manually pre-defined parameter is an obstacle for implementation of the feature-adjacency based ordering, since it's hard to determine the most suitable value for the parameter across different tasks. More generally, the feature-adjacency method depends on limited number of individual features, which normally needs very strong feature selection techniques to be effective. To avoid the sensitivity to individual features, we propose a cluster-adjacency based sentence ordering. Although the clustering will also use individual features, the noisy ones would be lower weighted via appropriate weighting schemes.

Assuming there are *n* summary sentences to be ordered, we cluster sentences in source documents into *n* clusters based on the *n* summary sentences. Each cluster represents a summary sentence. Then we use the cluster adjacency instead of feature adjacency to produce sentence adjacency. Since features are not directly used in calculating sentence

adjacency, the setting of the parameter to remove noisy features is no more needed. In addition, we expect the clustering to determine the themes properly and reduce the affect of noisy features.

3. Sentence Clustering

Assume there are K summary sentences to be ordered, and there are N sentences in source documents, we cluster the N sentences into K clusters using a semi-supervised classification method, Label Propagation (Zhu and Ghahramani, 2003). The advantage of this method is that it can exploit the closeness between unlabeled data during classification, thus ensuring a better classification result even with very fewer labeled data. This is exactly the situation here, where each summary sentence can be seen as the only one labeled data for the class.

Following are some notations for the label propagation algorithm in sentence classification:

$\{r_j\}$ $(1 \le j \le K)$: the K summary sentences

$\{m_j\}$ $(1 \le j \le N)$: the N document sentences to be classified

$X = \{x_i\}$ $(1 \le i \le K+N)$ refers to the union set of the above two categories of sentences, i.e. x_i $(1 \le i \le K)$ represents the K summary sentences, x_i $(K+1 \le i \le K+N+1)$ represents the N sentences to be classified. That is, the first K sentences are labeled sentences while the remaining N sentences are to be re-ranked. $C = \{c_j\}$ $(1 \le j \le K)$ denotes the class set of sentences, each one in which is labeled by a summary sentence. $Y^0 \in H^{s \times K}$ $(s=K+N)$ represents initial soft labels attached to each sentence, where $Y_{ij}^0 = 1$ if x_i is c_j and 0 otherwise. Let Y_L^0 be top $l=K$ rows of Y^0, which corresponds to the labeled data, and Y_U^0 be the remaining N rows, which corresponds to the unlabeled data. Here, each row in Y_U^0 is initialized according to the similarity of a sentence with the summary sentences.

In the label propagation algorithm, the manifold structure in X is represented as a connected graph and the label information of any vertex in the graph is propagated to nearby vertices through weighted edges until the propagation process converges. Here, each vertex corresponds to a sentence, and the edge between any two sentences x_i and x_j is weighted by w_{ij} to measure their similarity. Here w_{ij} is defined as follows: $w_{ij} = \exp(-d_{ij}^2 / \sigma^2)$ if $i \ne j$ and $w_{ii} = 0$ $(1 \le i,j \le l+u)$, where d_{ij} is the distance between x_i and x_j, and σ is a scale to control the transformation. In this paper, we set σ as the average distance between summary sentences. Moreover, the weight w_{ij} between two sentences x_i and x_j is transformed to a probability $t_{ij} = P(j \rightarrow i) = w_{ij} / (\sum_{k=1}^{s} w_{kj})$, where t_{ij} is the probability to propagate a label from sentence x_j to sentence x_i. In principle, larger weights between two sentences mean easy travel and similar labels between them according to the global consistency assumption applied in this algorithm. Finally, t_{ij} is normalized row by row as in (1), which is to maintain the class probability interpretation of Y. The $s \times s$ matrix is denoted as $\overline{T}$ as in (1).

During the label propagation process, the label distribution of the labeled data is clamped in each loop and acts like forces to push out labels through unlabeled data. With this push originates from labeled data, the label boundaries will be pushed much faster along edges with larger weights and settle in gaps along those with lower weights. Ideally, we can expect that w_{ij} across different classes should be as small as possible and w_{ij} within a same class as big as possible. In this way, label propagation happens within a same class most likely.

$$(1) \quad \overline{t_{ij}} = t_{ij} \Big/ \sum_{k=1}^{s} t_{ik}$$

$$(2) \quad \hat{Y}_U = \lim_{t \to \infty} Y_U^t = (I - \overline{T}_{uu})^{-1} \overline{T}_{ul} Y_L^0 .$$

$$(3) \quad \overline{T} = \begin{bmatrix} \overline{T}_{ll} & \overline{T}_{lu} \\ \overline{T}_{ul} & \overline{T}_{uu} \end{bmatrix}$$

This algorithm has been shown to converge to a unique solution (Zhu and Ghahramani, 2003) with $u=M$ and $l=K$ as in (2), where I is $u \times u$ identity matrix. $\overline{T}_{uu}$ and $\overline{T}_{ul}$ are acquired by splitting matrix $\overline{T}$ after the l-th row and the l-th column into 4 sub-matrices as in (3).

In theory, this solution can be obtained without iteration and the initialization of Y_U^0 is not important, since Y_U^0 does not affect the estimation of $\hat{Y}_U$. However, the initialization of Y_U^0 helps the algorithm converge quickly in practice. In this paper, each row in Y_U^0 is initialized according the similarity of a sentence with the summary sentences. Fig. 1 gives the classification procedure.

INPUT
 $\{x_i\}$ $(1 \le i \le K)$: set of summary sentences as labeled data;
 $\{x_i\}$ $(K+1 \le i \le K+N+1)$: set of document sentences;
 Algorithm: Label_Propagation($\{r_j\}$, $\{m_j\}$)
BEGIN
 Set the iteration index $t=0$
 BEGIN DO Loop
  Propagate the label by $Y^{t+1} = \overline{T} Y^t$;
  Clamp labeled data by replacing top l row of Y^{t+1} with Y_L^0
 END DO Loop when Y^t converges;
END

Fig. 1 Label propagation for sentence classification

The output of the classification is a set of sentence clusters, and the number of the clusters equals to the number of summary sentences. In each cluster, the members can be ordered by their membership probabilities. In fact, the semi-supervised classification is a kind of soft labeling (Tishby and Slonim, 2000; Zhou et al., 2003), in which each sentence belongs to different clusters, but with different probabilities. For sentence ordering task here, we need to get *hard* clusters, in which each sentence belongs to only one cluster. Thus, we need to cut the soft clusters to hard ones. To do that, for each cluster, we consider every sentence inside according to their decreasing order of their membership probabilities. If a sentence belongs to the current cluster with the highest probability, then it is selected and kept. The selection repeats until a sentence belongs to another cluster with higher probability.

4. Sentence Ordering

Given a set of summary sentences $\{S_1,\ldots,S_K\}$, sentences of the source documents are clustered into K groups $G_1,\ldots,G_K$,

where S_i is corresponding with G_i. For each pair of sentences S_i and S_j, the adjacency of S_i and S_j can be defined as the adjacency of G_i and G_j, defined in (4).

$$C_{i,j} = \frac{f(G_i, G_j)^2}{f(G_i) f(G_j)} \quad (4)$$

Here $f(G_i)$ and $f(G_j)$ respectively denote the frequency of cluster G_i and G_j in source documents, $f(G_i, G_j)$ denotes the frequency of G_i and G_j co-occurring in the source documents within a limited window range.

The first sentence S_1 can be determined according to (5) based on the adjacency between null clusters (containing only the null sentence) and any sentence clusters.

$$S_1 = \arg\max_{S_j \in T}(C_{o,j}) \quad (5)$$

Here $C_{0,j}$ denotes how close the sentence S_j and a null sentence are. By adding a null sentence at the beginning of each source document as S_0, and assuming it contains one null sentence, $C_{0,j}$ can be calculated with equation (4).

Given an already ordered sentence sequence, S_1, S_2,…,S_i, whose sentence set R is subset of the whole sentence set T, the task of finding the *(i+1)*th sentence can be described as:

$$S_{i+1} = \arg\max_{S_j \in T-R}(C_{i,j}) \quad (6)$$

Now the sentence sequence become S_1, S_2,…,S_i, S_{i+1}. By repeating the step the whole sequence can be derived.

5. Experiments and Evaluation

In this section, we describe the experiments with cluster-adjacency based ordering, and compared it with majority ordering, probability-based ordering and feature-adjacency based ordering respectively. Some methods [e.g., 8] tested ordering models using external training corpus and extracted sentence features such as nouns, verbs and dependencies from parsed tress. In this paper, we only used the raw input data, i.e., source input documents, and didn't use any grammatical knowledge. For feature-adjacency based model, we used single words except stop words as features to represent sentences. For cluster-adjacency based model, we used the same features to produce vector representations for sentences.

5.1 Test Set and Evaluation Metrics

Regarding test data, we used DUC04 data. DUC 04 provided 50 document sets and four manual summaries for each document set in its Task2. Each document set consists of 10 documents. Sentences of each summary were taken as inputs to ordering models, with original sequential information being neglected. The output ordering of various models were to be compared with that specified in manual summaries.

A number of metrics can be used to evaluate the difference between two orderings. In this paper, we used Kendall's τ [9], which is defined as:

$$\tau = 1 - \frac{2(number_of_inversions)}{N(N-1)/2} \quad (7)$$

Here N is the number of objects to be ordered (i.e., sentences). *Number_of_inversions* is the minimal number of interchanges of adjacent objects to transfer an ordering into another. Intuitively, τ can be considered as how easily an ordering can be transferred to another. The value of τ ranges from -1 to 1, where 1 denotes the best situation ---- two orderings are the same, and -1 denotes the worst situation--- completely converse orderings. Given a standard ordering, randomly produced orderings of the same objects would get an average τ of 0. For examples, Table 2 gives three number sequences, their natural sequences and the corresponding τ values.

Examples	Natural sequences	τ values
1 2 4 3	1 2 3 4	0.67
1 5 2 3 4	1 2 3 4 5	0.4
2 1 3	1 2 3	0.33

Table 2. Ordering Examples

5.2 Results

In the following, we used Rd, Mo, Pr, Fa and Ca to denote random ordering, majority ordering, probabilistic model, feature-adjacency based model and cluster-adjacency based model respectively. Normally, for Fa and Ca, the window size is set as 3 sentences, and for Fa, the noise elimination parameter (top-n) is set as 4.

Table 3 gives automatic evaluation results. We can see that Mo and Pr got very close τ values (0.143 *vs.* 0.144). Meanwhile, Fa got better results (0.316), and the Ca achieved the best performance (0.415). The significance tests suggest that the difference between the τ values of Fa and Mo or Pr is significant, and so is the difference between the values of Ca and Fa, where *, **, ~ represent p-values <=0.01, (0.01, 0.05], and >0.05.

Models	τ	**Significance**	**SVM**
Rd	-0.007		
Mo	0.143		0.153~
Pr	0.144		
Fa	0.316	**	
Ca	0.415	*	0.305**

Table 3. Automatic evaluation results

Both Mo and Ca use the themes acquired by the classification. In comparison, we also used SVM to do the classification, and Table 3 lists the τ values for Mo and Ca. SVM is a typical supervised classification, which only uses the comparison between labeled data and unlabeled data. So, it generally requires a large number of training data to be effective. The results show that the difference between the performance of Mo with LP (0.143) and SVM (0.153) is not significant, while the difference between the performance of Ca with LP (0.415) and SVM (0.305) is significant.

In general, if an ordering gets a positive τ value, the ordering can be considered to be better than a random one.

On the contrary, for a negative τ value, the ordering can be considered to be worse than a random one. For a zero τ value, the ordering is in fact close to a random one. So, percentage of τ values reflects quality of the orderings to some extent. Table 4 shows the percentage of positive ordering, negative orderings and median orderings for different models. It demonstrates that the cluster-adjacency based model produced the most positive orderings and the least negative orderings.

Models	**Positive Orderings**	**Negative Orderings**	**Median Orderings**
Rd	99 (49.5%)	90 (45.0%)	11 (5.5%)
Mo	123 (61.5%)	64 (32.0%)	13 (6.5%)
Pr	125 (62.5%)	59 (29.5%)	16 (8.0%)
Fa	143 (71.5%)	38 (19.0%)	19 (9.5%)
Ca	162 (81.0%)	31 (15.5%)	7 (3.5%)

Table 4. Positive, Negative and Median Orderings

To see why the cluster-adjacency model achieved better performance, we checked about the determination of the first sentence between different models, since that it is extremely important for Pr, Fa and Ca, and it will influence later selections. Either in Pr or in Fa and Ca, it was assumed that there is one null sentence at the beginning of each source document. In Pr, to determine the first sentence is to find one which is the most likely to follow the assumed null sentence, while in the two adjacency models, to determine the first sentence means to select one which is the closest to the null sentence. Table 5 shows the comparison.

Models	**Correct selection of 1st sentences**
Rd	22 (14.0%)
Mo	53 (26.5%)
Pr	81 (41.5%)
Fa	119 (59.5%)
Ca	131 (65.5%)

Table 5. First sentence determination

Table 5 indicates that cluster-adjacency model performed best in selection of the first sentence in the summaries.

Another experiment we did is about how likely the *k*+1th sentence can be correctly selected if assuming that top *k* sentences have been successfully acquired. This is also useful to explain why a model performs better than others. Fig. 2 shows the comparison of the probabilities of correct determination of the k+1th sentence between different models. Fig. 2 demonstrates that the probabilities of the correct k+1th sentence selection in cluster-adjacency model are generally higher than those in other methods, which indicates that the cluster-adjacency model is more appropriate for the data.

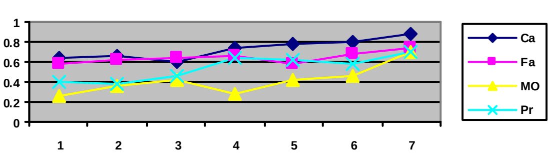


Fig. 2. k+1th sentence determination

Table 6 gives the experiment results of the cluster-adjacency model with varying window ranges. In general, the cluster-adjacency model got better performance than feature-adjacency model without requirement of setting the noise elimination parameters. This can be seen as an advantage of Ca over Fa. However, we can see that the adjacency window size still influenced the performance as it did for Fa.

Window size	τ values
2	0.314
3	0.415
4	0.398
5	0.356

Table 6. Ca performance with different window size

As a concrete example, consider a summary (D31050tG) in Fig. 3, which includes 6 sentences as the following.

0. After 2 years of wooing the West by signing international accords, apparently relaxing controls on free speech, and releasing and exiling three dissenters, China cracked down against political dissent in Dec 1998.
1. Leaders of the China Democracy Party (CDP) were arrested and three were sentenced to jail terms of 11 to 13 years.
2. The West, including the US, UK and Germany, reacted strongly.
3. Clinton's China policy of engagement was questioned.
4. China's Jiang Zemin stated economic reform is not a prelude to democracy and vowed to crush any challenges to the Communist Party or "social stability".
5. The CDP vowed to keep working, as more leaders awaited arrest.

Fig. 3. A sample summary

Table 7 gives the ordering generated by various models.

Models	Output	τ values
Pr	4 0 1 3 5 2	0.20
Mo	1 4 3 0 2 5	0.20
Fa	0 1 4 3 5 2	0.47
Ca	1 2 0 3 4 5	0.73

Table 7. Comparison: an example

From Table 7, we have several findings. First, sentence 3, 4 and 5 were close in the sequence in terms of their adjacency values, so in both Fa and Ca, once one of them was selected, the other two would follow. However, the closeness between them was not reflected in both Pr and Mo. Second, while Ca correctly made 1 followed by 2, Fa didn't. The reason may be that although sentence 1 and 2 had higher cluster-adjacency value, their feature-adjacency value may be lower than that between sentence 1 and 4, since sentence 1 and 4 shared more features, and only considering a limited number of features may make them get higher feature-adjacency value. At the same time, during classification in Ca, other different features (other than 'China', 'democracy', etc) would come to distinguish between sentence 1 and 4, so cluster centers of sentence 1 and 4 would have bias toward the distinguishing features. Thus, their adjacency value tended to be lower in Ca, and in fact, they fell apart in the sequence. Third, Fa successfully got the first sentence, while Ca didn't. To see the reason, we checked the summaries, and found that some summaries started with theme 0 and some more with theme 1, since theme 1 had part of the features in theme 0 and they may have contribution to feature-adjacency value, topic 1

tended to have higher feature-adjacency value. This is not contradicting with higher cluster-adjacency value between theme Null and theme 1. In fact, we found putting sentence 1 at the beginning was also acceptable subjectively.

In manual evaluation, the number of inversions was defined as the minimal number of interchanges of adjacent objects to transfer the output ordering to an acceptable ordering judged by human. We have three people participating in the evaluation, and the minimal, maximal and average numbers of interchanges for each summary among the three persons were selected for evaluation respectively. The Kendall's τ of all 5 runs are listed in Table 8.

	τ values		
Models	Average	Minimal	Maximal
Rd	0.106	0.202	0.034
Mo	0.453	0.543	0.345
Pr	0.465	0.524	0.336
Fa	0.597	0.654	0.423
Ca	0.665	0.723	0.457

Table 8. Manual evaluation results on 10 summaries

From table 7, we can find that all models get higher Kendall's τ values than in automatic evaluation, and the two adjacency models achieved better results than Pr and Mo according to the three measures. As example, Table 9 lists the subjective evaluation for the sample summary in Fig. 3.

Models	Output	Subjective ordering	τ values
Pr	4 0 1 3 5 2	401235	0.73
Mo	1 4 3 0 2 5	140235	0.73
Fa	0 1 4 3 5 2	014235	0.73
Ca	1 2 0 3 4 5	120345	1.0

Table 9. Subjective evaluation: an example

6. Conclusion and Future Work

In this paper we propose a cluster-adjacency based model for sentence ordering in multi-document summarization. It learns adjacency information of sentences from the source documents and order sentences accordingly. Compared with the feature-adjacency model, the cluster-adjacency method produces sentence adjacency from cluster adjacency. The major advantage of this method is that it focuses on a kind of global adjacency (cluster on the whole), and avoids sensitivity to limited number of features, which in general is difficult. In addition, with semi-supervised classification, this method is expected to determine appropriate themes in source documents and achieve better performance.

Although the cluster-adjacency based ordering model solved the problem of noise elimination required by the feature-adjacency based ordering, how to set another parameter properly, i.e., the window range, is still unclear. We guess it may depend on length of source documents. The longer the source documents are, the bigger adjacency window size may be expected. But more experiments are needed to prove it.

In addition, the adjacency based model mainly uses only adjacency information to order sentences. Although it appears to perform better than models using only sequential information on DUC2004 data set, if some sequential information could be learned definitely from the source documents, it should be better to combine the adjacency information and sequential information.

Reference

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2001. Sentence ordering in multidocument summarization. Proceedings of the First International Conference on Human Language Technology Research (HLT-01), San Diego, CA, 2001, pp. 149–156..

Barzilay, R N. Elhadad, and K. McKeown. 2002. *Inferring strategies for sentence ordering in multidocument news summarization.* Journal of Artificial Intelligence Research, 17:35–55.

Sasha Blair-Goldensohn, David Evans. Columbia University at DUC 2004. In Proceedings of the 4th Document Understanding Conference (DUC 2004). May, 2004.

Danushka Bollegala, Naoaki Okazaki, Mitsuru Ishizuka. 2005. A machine learning approach to sentence ordering for multidocument summarization and it's evaluation. IJCNLP 2005, LNAI 3651, pages 624-635, 2005.

McKeown K., Barzilay R. Evans D., Hatzivassiloglou V., Kan M., Schiffman B., &Teufel, S. (2001). *Columbia multi-document summarization: Approach and evaluation.* In Proceedings of DUC.

Mirella Lapata. Probabilistic text structuring: Experiments with sentence ordering. Proceedings of the annual meeting of ACL, 2003., pages 545–552, 2003.

Nie Yu, Ji Donghong and Yang Lingpeng. An adjacency model for sentence ordering in multi-document Asian Information Retrieval Symposium (AIRS2006), Singapore., Oct. 2006.

Advaith Siddharthan, Ani Nenkova and Kathleen McKeown. Syntactic Simplication for Improving Content Selection in Multi-Document Summarization. In Proceeding of COLING 2004, Geneva, Switzerland.

Tishby, N, Slonim, N. (2000) *Data clustering by Markovian relaxation and the Information Bottleneck Method*. NIPS 13.

Szummer M. and T. Jaakkola. (2001) *Partially labeled classification with markov random walks.* NIPS14.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003) *Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions*. ICML-2003.

Zhou D., Bousquet, O., Lal, T.N., Weston J. & Schokopf B. (2003). *Learning with local and Global Consistency*. NIPS 16. pp: 321-328

Statistical Machine Translation based Passage Retrieval for Cross-Lingual Question Answering

Tomoyosi Akiba Kei Shimizu
Dept. of Information and Computer Sciences,
Toyohashi University of Technology
1-1 Hibarigaoka, Tenpaku-cho, Toyohashi-shi,
441-8580, JAPAN
akiba@cl.ics.tut.ac.jp

Atsushi Fujii
Graduate School of Library,
Information and Media Studies,
University of Tsukuba
1-2 Kasuga, Tsukuba, 305-8550, JAPAN
fujii@slis.tsukuba.ac.jp

Abstract

In this paper, we propose a novel approach for Cross-Lingual Question Answering (CLQA). In the proposed method, the statistical machine translation (SMT) is deeply incorporated into the question answering process, instead of using it as the pre-processing of the mono-lingual QA process as in the previous work. The proposed method can be considered as exploiting the SMT-based passage retrieval for CLQA task. We applied our method to the English-to-Japanese CLQA system and evaluated the performance by using NTCIR CLQA 1 and 2 test collections. The result showed that the proposed method outperformed the previous pre-translation approach.

1 Introduction

Open-domain Question Answering (QA) was first evaluated extensively at TREC-8 (Voorhees and Tice, 1999). The goal in the factoid QA task is to extract words or phrases as the answer to a question from an unorganized document collection, rather than the document lists obtained by traditional information retrieval (IR) systems. The cross-lingual QA task, which has been evaluated at CLEF (Magnini et al., 2003) and NTCIR (Sasaki et al., 2005), generalizes the factoid QA task by allowing the different languages pair between the question and the answer.

Basically, the CLQA system can be constructed simply by translating either the question sentence or the target documents into the language of the other side, and applying a mono-lingual QA system. For example, after the English question sentence is translated into Japanese, a Japanese mono-lingual QA system can be applied to extract the answer from the Japanese target documents. Depending on the translation techniques used for the pre-processing, the previous CLQA approach can be classified into the machine translation based approach (Shimizu et al., 2005; Mori and Kawagishi, 2005) and the dictionary based approach (Isozaki et al., 2005).

In this paper, we propose a novel approach for CLQA task. In the proposed method, the statistical machine translation (SMT) (Brown et al., 1993) is deeply incorporated into the question answering process, instead of using the SMT as the pre-processing before the mono-lingual QA process as in the previous work. Though the proposed method can be applied to any language pairs in principle, we focus on the English-to-Japanese (EJ) CLQA task, where a question sentence is given in English and its answer is extracted from a document collection in Japanese.

Recently, language modeling approach for information retrieval has been widely studied (Croft and Lafferty, 2003). Among them, statistical translation model has been applied for mono-lingual IR (Berger and Lafferty, 1999), cross-lingual IR (Xu et al., 2001), and mono-lingual QA (Murdock and Croft, 2004). Our method can be considered as that applying the translation model to cross-lingual QA.

In the rest of this paper, Section 2 summarizes the previous approach for CLQA. Section 3 describes our proposed method in detail. Section 4 describes the experimental evaluation conducted to see the performance of the proposed method by comparing it to some reference methods. Section 5 describes our conclusion and future works.

2 Previous CLQA Systems

Figure 1 shows the configuration of our previous English-to-Japanese cross-lingual QA system, which has almost the same configuration to the conventional CLQA systems. Firstly, the input English question is translated into the corresponding Japanese question by using a machine translation. Alternatively, the machine translation can be re-

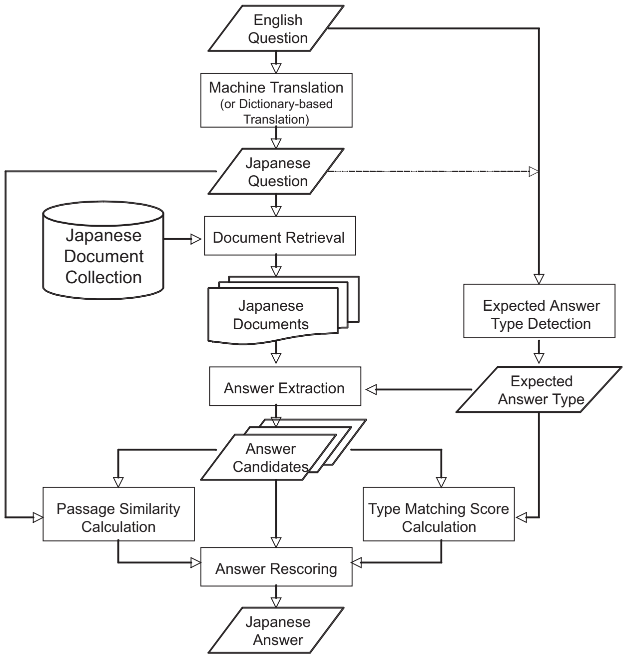


Figure 1: The configuration of the conventional CLQA system.

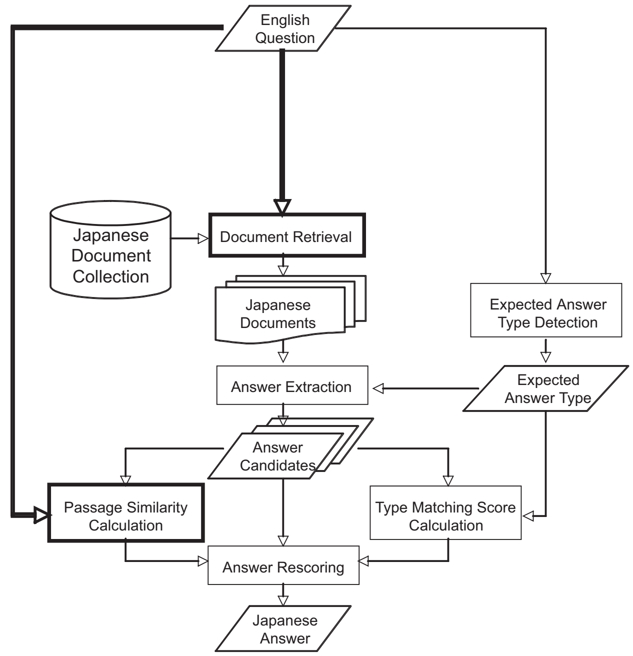


Figure 2: The configuration of the proposed CLQA system.

placed by the dictionary-based term-by-term translation. Then, either the English question or the translated Japanese question is analyzed to get the expected answer type.

After that, the mono-lingual QA process is invoked. The translated Japanese question is used as the query of the document retrieval to get the documents that include the query terms. From the retrieved documents, the answer candidates that match with the expected answer type are extracted with their location in the documents. Next, the extracted candidates are rescored by the two points of views; the passage similarity and the type matching. The passage similarity is calculated between the translated Japanese question and the Japanese passage that surrounds the answer candidate, while the type matching score is calculated as the likelihood that the candidate is matched with the expected answer type. Finally the reordered candidates are outputted as the answers of the given question.

3 Proposed CLQA System

On the other hand, Figure 2 shows the configuration of our proposed cross-lingual QA system. It does not use the machine translation (nor the dictionary-based translation) as the pre-processing of the input English question. The original English question is used directly in the QA process. In order to make this approach possible, the two subsystems, the document retrieval subsystem and the passage similarity calculation subsystem, which are pointed by the direct arrow from the English question and are emphasized by the thick frames in Figure 2, are *cross-lingualized* to accept the English question directly instead of the Japanese question, by means of incorporating the statistical machine translation (SMT) process deeply into them.

In the following two subsections, we will explain how these two subsystems can deal with the English question directly. The document retrieval subsystem is modified so that the Japanese documents are indexed by English terms. The word translation probability used in the SMT is used to index the Japanese document with the corresponding English terms without losing the consistency. The passage similarity calculation subsystem calculates the similarity between an English question and a Japanese passage in terms of the probability that the Japanese passage is translated into the English question.

3.1 Document Retrieval

Given an English question sentence, the document retrieval subsystem of our proposed CLQA system retrieves Japanese documents directly. In order to do so, each Japanese document in the target collection

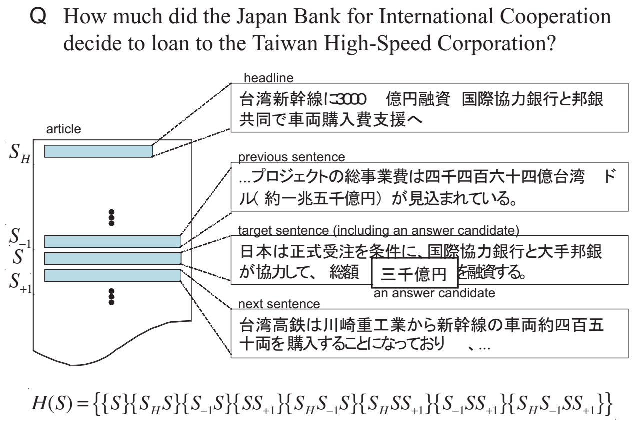


Figure 3: An examples of a question and the corresponding passage candidates.

has been indexed by English terms by using the word translation probability used in the SMT framework.

The expected term frequency $tf(e, D)$ of an English term e that would be used as an index to a Japanese document D can be estimated by the following equation.

$$tf(e, D) = \sum_{j \in D} t(e|j) tf(j, D) \quad (1)$$

where $tf(j, D)$ is the term frequency of a Japanese term j in D and $t(e|j)$ is the word translation probability that j is translated into e. The probability $t(e|j)$ is trained by using a large parallel corpus as the SMT framework. Because the expected term frequency $tf(e, D)$ is consistent with $tf(j, D)$ that is calculated from the statistics of D, the conventional vector space IR model based on the TF-IDF term weighting can be used for implementing our IR subsystem. We used *GETA* [1] as the IR engine in our CLQA system.

3.2 SMT based Passage Retrieval

In order to enable the direct passage retrieval, where the query and the passage are in different languages, the statistical machine translation is utilized to calculate the similarity between them. In order words, we calculate the similarity between them as the probability that the Japanese passage is translated into the English question.

The similarity $sim(Q, S|A)$ between a question Q and a sentence S including an answer candidate A is calculated by the following equation.

$$sim(Q, S|A) = \max_{D \in H(S)} P(Q|D - A) \quad (2)$$

[1] http://geta.ex.nii.ac.jp

where $P(Q|D - A)$ is the probability that a word sequence D except A is translated into a question sentence Q, and $H(S)$ is the set of the candidate passage (term sequences) that are related to a sentence S. The set consists of S and the power set of S_H, S_{-1}, and S_{+1}, where S_H is the headline of the article that S belongs, S_{-1} is the previous sentence of S, and S_{+1} is the next sentence of S (Figure 3).

In this paper, we use IBM model 1 (Brown et al., 1993) in order to get the probability $P(Q|D - A)$ as follows.

$$P(Q|D - A) = \frac{1}{(n+1)^m} \prod_{j=1}^{m} \sum_{i=1,\cdots,k-1,k+l+1,\cdots,n} t(q_j|d_i) \quad (3)$$

where $q_1 \cdots q_m$ is a English term sequence of a question Q, $d_1 \cdots d_n$ is a Japanese term sequence of a candidate passage D, $d_k \cdots d_{k+l}$ is a Japanese term sequence of an answer candidate A. Therefore, the Japanese term sequence $d_1, \cdots, d_{k-1}, d_{k+l+1}, \cdots, d_n$ (= D - A) is just D except A. We exclude the answer term sequence A from the calculation of the translation probability, because the English terms that corresponds to the answer should not be appeared in the question sentence as the nature of question answering.

4 Experimental Evaluation

The experimental evaluation was conducted to see the total performance of cross language question answering by using our proposed method.

4.1 Test collections

The NTCIR-5 CLQA1 test collection (Sasaki et al., 2005) and the NTCIR-6 CLQA2 test collection (Sasaki et al., 2007) for English-to-Japanese task were used for the evaluation. Each collection contains 200 factoid questions in English. The target documents for CLQA1 are two years newspaper articles from "YOMIURI SHINBUN" (2000-2001), while those for CLQA2 are two years articles from "MAINICHI SHINBUN" (1998-1999).

In the test collections, the answer candidates are judged with three categories; **Right**, **Unsupported**, and **Wrong**. The answer labeled **Right** is correct and supported by the document that it is from. The answer labeled **Unsupported** is correct but not supported by the document that it is from. The answer labeled **Wrong** is incorrect. We used two kind of golden set for our evaluation: the set including only

Right answers (referred as to **R**) and the set including **Right** and **Unsupported** answers (referred as to **R+U**).

Note that the evaluation results obtained from CLQA2 are more reliable than that from CLQA1, because we participated in CLQA2 formal run with our proposed method (and our reference method labeled **DICT**) and most of the answers by the system were manually checked for the pooling.

4.2 Translation Model

The translation model used for our method was trained from the following English-Japanese parallel corpus.

- 170,379 example sentence pairs from the Japanese-English and English-Japanese dictionaries.
- 171,186 sentence pairs from newspaper articles obtained by the automatic sentence alignment (Utiyama and hitoshi Isahara, 2003).

A part of the latter sentence pairs were obtained from the paired newspapers that are "YOMIURI SHINBUN" and its English translation "Daily Yomiuri". Because the target documents of CLQA1 are the articles from "YOMIURI SHINBUN" as described above, the corresponding sentence pairs, which are extracted from the articles from 2000 to 2001, were removed from the training corpus for CLQA1.

Before training the translation model, both English and Japanese sides of the sentence pairs in parallel corpus were normalized. For the sentences of Japanese side, the inflectional words were normalized to their basic forms by using a Japanese morphological analyzer. For the sentences of English side, the inflectional words were also normalized to their basic forms by using a Part-of-Speech tagger and all the words were lowercased. GIZA++ (Och and Ney, 2003) was used for training the IBM model 4 from the normalized parallel corpus. The vocabulary sizes were about 58K words for Japanese side and 74K words for English side. The trained Japanese-to-English word translation model $t(e|j)$ was used for our proposed document retrieval (Section 3.1) and passage similarity calculation (Section 3.2).

4.3 Compared methods

The proposed method was compared with the several reference methods. As the methods from previous works, three pre-translation methods were investigated.

The first two methods translate the question by using machine translation. One of them used a commercial off-the-shell machine translation software [2] (referred to as **RMT**). The other used the statistical machine translation that had been created by using the IBM model 4 obtained from the same parallel corpus and tools described in Section 4.2, the tri-gram language model constructed by using the target documents of CLQA1, and the existing SMT decoder (Germann, 2003) (referred to as **SMT**). The two methods, **RMT** and **SMT**, differ only in the translation methods, while their backend monolingual QA systems are common.

The third method translates the question by using translation dictionary (referred to as **DICT**). The cross-lingual IR system described in (Fujii and Ishikawa, 2001) was used for our "document retrieval" subsystem in Figure 2. The CLIR system enhances the basic translation dictionary, which has about 1,000,000 entries, with the compound words obtained by using the statistics of the target documents and with the borrowed words by using the transliteration method. Note that, as the other parts of the system than the document retrieval, including proposed SMT based passage retrieval, are all identical to the proposed method, this comparison is focused only on the difference in the document retrieval methods.

In order to investigate the performance if the ideal translation is made, the reference Japanese translations of the English questions included in the test collections were used as the input of the monolingual QA system (referred to as **JJ**).

As the variations of the proposed method, the following four methods were compared.

Proposed The same method as described in Section 3.

Proposed +r The document retrieval score is also used to rescore the answer candidates in "Rescoring" subsystem in Figure 2, in addition to the passage similarity score and the type matching score.

Proposed -p For the passage similarity calculation, the passage is always fixed only the central sentence S, i.e. the equation (2) is replaced by the following.

$$sim(Q, S|A) = P(Q|S - A) \qquad (4)$$

Proposed -p+r Combination of above two modifications.

[2] "IBM Japan, honyaku-no-oosama ver. 5"

Table 1: Comparison of the **JJ** results between the test collections.

test collection	R			R+U		
	Top1 Acc.	Top5 Acc.	MRR	Top1 Acc.	Top5 Acc.	MRR
CLQA1	0.140	0.300	0.196	0.260	0.535	0.354
CLQA2	0.245	0.410	0.307	0.270	0.530	0.366

Table 2: The performances of the proposed and reference CLQA systems with respect to CLQA1 test collection.

method	Top1 Acc.	Top5 Acc.	MRR
RMT	0.065	0.175	0.099
SMT	0.060	0.175	0.098
Dict	0.095	0.195	0.134
Proposed	0.090	0.225	0.146

Table 3: The performances among the proposed methods with respect to CLQA1 test collection.

method	Top1 Acc.	Top5 Acc.	MRR
Proposed	0.090	0.225	0.146
Proposed +r	0.105	0.285	0.173
Proposed -p	0.105	0.245	0.155
Proposed -p+r	0.120	0.280	0.178
JJ	0.260	0.535	0.354

4.4 Evaluation Metrics

Each system outputted five ranked answers $a_1 \cdots a_5$ for each question q. We investigated the performance of the systems in terms of three evaluation metrics that are obtained by averaging over all the questions: the accuracy of the top ranked answers (referred to as **Top 1 Acc.**), the accuracy of up-to fifth ranked answers (referred to as **Top 5 Acc.**), and the reciprocal rank (referred to as **MRR**) $RR(q)$ calculated by the following equation.

$$rr(a_i) = \begin{cases} 1/i & \text{if } a_i \text{ is a correct answer} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$RR(q) = \max_{a_i} rr(a_i) \quad (6)$$

4.5 Results

Firstly, we compared the results obtained by using CLQA1 test collection with that obtained by using CLQA2. Table 1 shows the results for **JJ** system. By using the **R** judgment, the **JJ** results of CLQA1 was much worse than that of CLQA2, while the results were almost same by using the **R+U** judgment. Because the difference with respect to the difficulties between the two test collections seems small and the results from CLQA2 are more reliable, we concluded that the **R** judgment of CLQA1 was unreliable. Therefore, for CLQA1 test collection, we only investigated the result by using **R+U** judgment.

Secondly, we compared the proposed method (**Proposed**) with the previous methods (**RMT**, **SMT**, and **Dict**). Table 2 shows the results with respect to CLQA1 test collection. The two methods based on the machine translation (**RMT** and **SMT**) indicated almost same performance, while the performance of the proposed method was about 1.3 to 1.5 times better for CLQA1. Especially, because the same training data was used to build the translation models both in **SMT** and **Proposed**, it was shown that the method to build the SMT model in the QA process was better than that to use the same SMT model for pre-processing (pre-translating) the input sentence.

The **DICT** performed almost same as the **Proposed** for CLQA1, while **Proposed** was 1.7 to 1.9 times better than **DICT** for CLQA2 as shown in Table 4. Note again that this comparison was focused on the document retrieval subsystem, because the passage retrieval subsystems of these two methods were same.

Thirdly, the variations between the proposed methods were compared. Table 3 shows the results with respect to CLQA1 test collection. For CLQA1, both the additional use of the document retrieval score (**+r**) and the use of the fixed central sentence for passage similarity calculation (**-p**) improved the performance. However, for CLQA2, the document retrieval score (**+r**) did not contribute to improve the performance, as shown in Table 4.

Finally, seeing from the comparison between **JJ** and **Proposed**, it was shown that the performance of the proposed CLQA system was about half of that of the ideal CLQA system.

Table 4: The performances of the proposed and reference CLQA systems with respect to CLQA2 test collection.

methods	R			R+U		
	Top1 Acc.	Top5 Acc.	MRR	Top1 Acc.	Top5 Acc.	MRR
Dict	0.070	0.155	0.102	0.100	0.275	0.163
Proposed	0.130	0.200	0.155	0.165	0.295	0.210
Proposed +r	0.120	0.220	0.153	0.155	0.325	0.211
JJ	0.245	0.410	0.307	0.270	0.530	0.366

5 Conclusion

In this paper, a novel approach for CLQA was proposed. The proposed method did not translate the input question in source language into the target language as the preprocessing of QA process. Instead, the statistical machine translation was deeply incorporated into the two QA subsystems in order to deal with the question in source language directly in the QA process. Especially, SMT-based passage retrieval was explored.

For the passage similarity calculation in this paper, the simple IBM model 1 was used. In the future work, we will investigate if the more sophisticated translation model or that specialized for CLQA task can improve the performance further.

References

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22nd Annual Conference on Research and Development in Information Retrieval (ACM SIGIR)*, pages 222–229.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 18(4):263–311.

W. Bruce Croft and John Lafferty, editors. 2003. *Language Modeling for Information Retrieval*. Kluwer Academic Publishers.

Atsushi Fujii and Tetsuya Ishikawa. 2001. Japanese/english cross-language information retrieval: Exploration of query translation and transliteration. *Computers and the Humanities*, 35(4):389–420.

Ulrich Germann. 2003. Greedy decoding for statistical machine translatioin in slmost linear time. In *Proceedings of HLT-NAACL*.

Hideki Isozaki, Katsuhito Sudoh, and Hajime Tsukada. 2005. NTT's japanese-english cross-language question answering system. In *Proceedings of The Fifth NTCIR Workshop*, pages 186–193.

Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Pe nas, Maarten de Rijke, Paulo Rocha, Kiril Simov, and Richard Sutcliffe. 2003. Overview of the CLEF 2004 multilingual question answering track. In *Multilingual Information Access for Text, Speech and Images*, pages 371–391.

Tatsunori Mori and Masami Kawagishi. 2005. A method of cross language question-answering based on machine translation and transliteration. In *Proceedings of The Fifth NTCIR Workshop*, pages 215–222.

Vanessa Murdock and W. Bruce Croft. 2004. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of the Workshop on Information Retrieval for Question Answering*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Yutaka Sasaki, Hsin-Hsi Chen, Kuang hua Chen, and Chuan-Jie Lin. 2005. Overview of the NTCIR-5 cross-lingual question answering task (clqa1). In *Proceedings of The Fifth NTCIR Workshop*, pages 175–185.

Yutaka Sasaki, Chuan-Jie Lin, Kuang hua Chen, and Hsin-Hsi Chen. 2007. Overview of the NTCIR-6 cross-lingual question answering (clqa) task. In *Proceedings of The NTCIR-6 Workshop Meeting*.

Kei Shimizu, Tomoyosi Akiba, Atsushi Fujii, and Katunobu Itou. 2005. Bi-directional cross language question answering using a single monolingual QA system. In *Proceedings of The Fifth NTCIR Workshop*, pages 236–237.

Masao Utiyama and hitoshi Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 72–79.

E. Voorhees and D. Tice. 1999. The TREC-8 question answering track evaluation. In *Proceedings of the 8th Text Retrieval Conference*, pages 83–106, Gaithersburg, Maryland.

Jinxi Xu, Ralph Weischedel, and Chanh Nguyen. 2001. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the 24th Annual Conference on Research and Development in Information Retrieval (ACM SIGIR)*, pages 105–110.

Unsupervised All-words Word Sense Disambiguation with Grammatical Dependencies

Vivi Nastase
EML Research gGmbH
Heidelberg, 69118, Germany
nastase@eml-research.de

Abstract

We present experiments that analyze the necessity of using a highly interconnected word/sense graph for unsupervised all-words word sense disambiguation. We show that allowing only grammatically related words to influence each other's senses leads to disambiguation results on a par with the best graph-based systems, while greatly reducing the computation load. We also compare two methods for computing selectional preferences between the senses of every two grammatically related words: one using a Lesk-based measure on WordNet, the other using dependency relations from the British National Corpus. The best configuration uses the syntactically-constrained graph, selectional preferences computed from the corpus and a PageRank tie-breaking algorithm. We especially note good performance when disambiguating verbs with grammatically constrained links.

1 Introduction

It has long been believed that being able to detect the correct sense of a word in a given context – performing word sense disambiguation (WSD) – will lead to improved performance of systems tackling high end applications such as machine translation (Chan et al., 2007) and summarization(Elhadad et al., 1997). In order for WSD methods to be useful, they must be robust, portable, scalable, and therefore preferably not reliant on manually tagged data. These desiderata have lead to an increased interest in developing unsupervised WSD methods, flexible relative to the word sense inventory, and which disambiguate all open-class words in a given context as opposed to a selected few.

Particularly appropriate from this point of view are graph-based methods (Navigli and Lapata, 2007), which map the open-class words in a given context onto a highly interconnected graph. Each node in this graph represents a word sense, and weighted edges will connect every pair of senses (corresponding to different words). The topology of the graph and the weights of the edges can contribute in a variety of ways to determine the best sense combination for the words in the considered context. This approach leads to large and highly interconnected graphs, in which distant, unrelated (in the context) words, are nonetheless connected, and allowed to influence each other's sense preferences. We study the impact on disambiguation performance when connections are restricted to pairs of word senses corresponding to words that are grammatically linked in the considered context.

The benefits of using grammatical information for automatic WSD were first explored by Yarowsky (1995) and Resnik (1996), in unsupervised approaches to disambiguating single words in context.

Sussna (1993) presents a first approach to disambiguating together words within a context. The focus is on nouns, and the sense combination that minimizes the overall distance in the WordNet nouns network is chosen.

Stetina et al. (1998) present the first approach, supervised, to disambiguating all words in a sentence with sense association (or selectional) preferences computed from a sense-tagged corpus. An untagged grammatically linked word pair will have associated a matrix of sense combination scores, based on the analyzed sense-tagged corpus, and similarities between the current words and those in tagged pairs with the same grammatical relation. Once such matrices are computed for all grammatically related word pairs, the sense preferences are propagated from the bottom of the parse tree towards the top, and the sense selection starts from the top and propagates downward.

McCarthy and Carroll (2003) also use an unsupervised approach and grammatical relations to learn selectional preferences for word classes. In an approach inspired by the works of Li and Abe (1998) and Clark and Weir (2002), McCarthy and Carroll use grammatically connected words from a corpus to induce a distribution of senses over subtrees in the WordNet hierarchy. McCarthy et al. (2004) use a corpus and word similarities to induce a ranking of word senses from an untagged corpus to be used in WSD.

We build upon this previous research, and propose an unsupervised WSD method in which senses for two grammatically related words in the sentence will be connected through directed edges. We experiment with graph edge weights computed using WordNet, and weights computed using grammatical collocation information from a corpus. These

weights are used to induce an initial scoring of the graph vertices, starting from the leaves and propagating upwards. The disambiguation process starts with choosing a sense for the head of the sentence, and moves towards the leaves, propagating downward the chosen senses at each step, and using the edge weights and vertex scores to guide the sense selection process.

We investigate two issues: (i) whether using in disambiguation only syntactically connected words leads to results on a par with, or better than, using all word-sense combinations, (ii) whether sense association strength induced from a sense-unlabeled corpus can rival relatedness measures induced from a lexical resource - in our case, WordNet.

We evaluate this approach on the Senseval-2(Palmer et al., 2001) and Senseval-3(Snyder and Palmer, 2004) English all-words test data. On the Senseval-2 data we obtain results on a par with the best unsupervised WSD systems, on the Senseval-3 data, the results are lower overall, but for verbs higher than those obtained with other graph-based methods. In both situations, using only grammatically motivated edges leads to improved disambiguation of verbs compared to disambiguating in a graph with unrestricted connections.

2 Disambiguation Algorithm

The disambiguation method described here uses grammatical information from the sentential context to constrain word pairs that are allowed to influence each other's sense choice. Edge weights in the graph are relatedness scores computed based on WordNet and, in a second set-up, selectional preferences estimated from an (sense-)untagged corpus, for disambiguating together all words in the sentence. Grammatical information for the sentential context is obtained using the dependency relation output of the Stanford Parser (de Marneffe et al., 2006). Selectional preferences are estimated using grammatical collocation information from the British National Corpus (BNC), obtained with the Word Sketch Engine (WSE) (Kilgarriff et al., 2004).

2.1 Extracting grammatical relation information

We parse the Senseval test data using the Stanford Parser(Klein and Manning, 2003) generating the output in dependency relation format (de Marneffe et al., 2006). Edges that do not connect open-class words are filtered out, words are lemmatized, and we reintroduce the copula (it is bypassed as a predicate) because the verb *be* must be disambiguated as well.

To estimate selectional preferences from a sense-untagged corpus, for each grammatically related pair of words in a sentence we extract evidence consisting of pairs with the same grammatical relation and either the same head or dependent, using the Word Sketch Engine. To obtain such pairs we map the grammatical relations used by the Stanford Parser onto the set of grammatical relations used by the WSE. Table 1 shows a sample of this mapping. We denote by $GR\ ^{1}$ the inverse of grammatical relation GR – for example *subject_of* is the inverse of *subject*.

Dependency relation	WSE relation
nsubj(verb,noun)	subject(verb,noun) subject_of(noun,verb)
dobj(verb,noun)	object(verb,noun) object_of(noun,verb)
amod(noun,adj)	a_modifier(noun,adj) modifies(adj,noun)
nn($noun_1$,$noun_2$)	n_modifier($noun_1$,$noun_2$) modifies($noun_2$,$noun_1$)
prep_of(verb,noun)	pp_of(verb,noun) pp-obj_of(noun,verb)

Table 1: Mapping of grammatical relations from the Stanford Parser onto the WSE relation set – a sample.

The result of this processing is illustrated in Figure 1, for the following sentence from the Senseval2 test data:

The art of change-ringing is peculiar to the English, and, like most English peculiarities, unintelligible to the rest of the world.

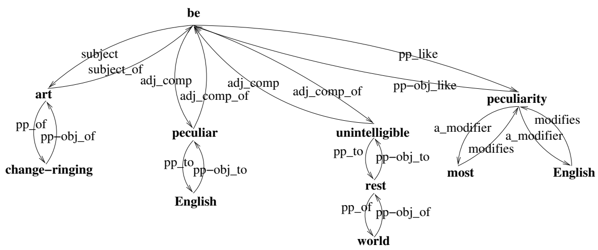


Figure 1: Dependency graph with grammatical relations mapped onto the WSE set

The dependency between two connected words is represented by two asymmetric grammatical relations.

2.2 Computing sense selectional preference scores

The selectional preference scores can be computed using the lexical resource that provides the inventory of senses, or using a corpus.

Sense-selectional preferences based on dependency relations in a corpus For each pair of words in a grammatical relation (w_1, w_2, GR) from a sentence, we compute a score for each sense s^i_{w2} of w_2, that shows the strength of the association

between s^i_{w2} and w_1. The strength of the association will come from collocation information from the BNC, combined with sense similarity or relatedness between $s^i_{w_2}$ and collocates of w_1 in grammatical relation GR.

Let us take an example – *(rest,world,pp_of)* from the example sentence presented before. We want to estimate the preferences of *rest* for senses of *world*. *world* has the following senses in WordNet 1.7[1]: world%1:17:02::[2], world%1:17:00::, world%1:17:01::, world%1:14:02::, world%1:14:01::, world%1:14:00::, world%1:09:01::, world%1:09:00:: .

From the BNC we obtain the following collocation information (the formatting of the list is w_1-POS GR w_x-POS:co-occurrence frequency):

rest-n pp_of life-n:639, world-n:518, Europe-n:211, cast-n:44, season-n:90, day-n:253, country-n:158, family-n:134, evening-n:60, Kingdom-n:42, chapter-n:55, team-n:96, week-n:93, society-n:89, afternoon-n:34, population-n:56, ...

The list of grammatical collocates with $rest$ in relation *pp_of* are: $GC^{pp_of}_{rest}$ = { *life, world, Europe, case, season, day, country, family, evening, Kingdom, chapter, team, week, society, afternoon, population,...* }

Based on relatedness scores between senses of these collocates and senses of *world* we compute selectional preference scores for each of *world*'s senses:

world%1:17:02::→1	world%1:17:00::→2
world%1:17:01::→3	world%1:14:02::→2
world%1:14:01::→3	world%1:14:00::→4
world%1:09:01::→1	world%1:09:00::→1

The same procedure is applied to compute the sense selectional preference scores of $world$ for each of $rest$'s senses, in the grammatical relation *pp-obj_of* (the inverse of *pp_of* in WSE).

Formally, for the tuple (w_1, w_2, GR), we extract from the BNC all pairs (w_1, w_x, GR)[3]. The set

$$GC^{GR}_{w_1} = \{w_x | (w_1, w_x, GR) \in \text{corpus}\}$$

gives w_1's grammatical collocations. To estimate the sense association strength between w_1 and senses of w_2, for each $w_x \in GC^{GR}_{w_1}$ we compute relatedness between the senses of w_x and the senses of w_2. $A^{s^i_{w_2}}_{w_1|GR}$, the association strength between w_1 and sense $s^i_{w_2}$ of word w_2 under relation GR, is the sum of these relatedness scores:

$$A^{s^i_{w_2}}_{w_1|GR} = \sum_{w_x \in GC^{GR}_{w_1}} \sum_{s^j_{w_x} \in S_{w_x}} rel(s^i_{w_2}, s^j_{w_x})$$

where S_{w_x} is the set of senses for word w_x.

If this value is 0, then $A^{s^i_{w_2}}_{w_1|GR} = \frac{1}{n_{w_2}}$, where n_{w_2} is the number of senses of w_2.

$rel(s^i_{w_2}, s^j_{w_x})$ can be computed as a similarity or relatedness measure (Budanitsky and Hirst, 2006). Because the sense inventory for the Senseval data comes from WordNet and we work at the sense level, we use relatedness measures based on WordNet, as opposed to corpus-based ones. In the experiments presented further in the paper, we have used a relatedness measure based on hypernym and hyponym information, in the following manner:

$$rel(s^i_{w_2}, s^j_{w_x}) = \begin{cases} 1 & : \; s^i_{w_2} \text{ is a hypernym of } s^j_{w_x} \\ 1 & : \; s^i_{w_2} \text{ is a hyponym of } s^j_{w_x} \\ & \quad \text{and } path_length(s^i_{w_2}, s^j_{w_x}) \leq 2 \\ 1 & : \; s^i_{w_2} \text{ similar to/antonym of } s^j_{w_x} \\ 0 & : \; otherwise \end{cases}$$

In other words, if the sense $s^i_{w_2}$ of w_2 is a hypernym of the sense $s^j_{w_x}$ or a close hyponym (distance at most 2) or connected through a *similar to/antonym of* relation, we consider the two senses related and relatedness gets a score of 1. Otherwise, we consider the two senses unrelated.

The motivation for using this relatedness measure is that it allows fast computations – essential when dealing with a large amount of information from a corpus – and it clusters closely related senses based on WordNet's hypernym/hyponym relations. By clustering together related senses, we gather more evidence for the selectional preferences of w_2's senses, which also helps partly with the data sparseness problem.

Because at this point it is not determined to which of w_x's senses the selectional preference is due, all of w_x's senses will have the same selectional preference to a sense j of w_y: $A^{s^j_{w_y}}_{s^i_{w_x}|GR} = A^{s^j_{w_y}}_{w_x|GR}$, for all senses $s^i_{w_x}$ of w_x.

Sense-selectional preferences based on a lexical resource When using the lexical resource, because we have pairs that connect words under different parts of speech, we opt for a Lesk-based measure (Banerjee and Pedersen, 2003). Relatedness scores are computed for each pair of senses of the grammatically linked pair of words (w_1, w_2, GR), using the WordNet-Similarity-1.03 package and the `lesk`

[1]WordNet 1.7 is the sense inventory for Senseval2, WordNet 1.7.1 is the sense inventory for Senseval 3.

[2]Unique sense identifier from the WN lexicographer files.

[3]Only w_x collocates that have the same part of speech as w_2 are considered.

option (Pedersen et al., 2004). To maintain the notation from above, we denote by $A_{s^j_{w_y}}^{s^i_{w_x}}$ the lesk relatedness score between sense i of w_x and sense j of w_y. These scores are symmetric: $A_{s^j_{w_y}}^{s^i_{w_x}} = A_{s^i_{w_x}}^{s^j_{w_y}}$, and independent of grammatical relations GR.

2.3 The sense-enhanced dependency tree

After computing the sense association strength scores for w_1 and w_2 in grammatical relation GR in the sentence, we expand the edge (w_x, w_y, GR) from the dependency tree to the two sets of directed edges:

$$\{(s^i_{w_x} \rightarrow s^j_{w_y}, GR) | i = 1, n; j = 1, m\},$$
$$\{(s^j_{w_y} \rightarrow s^i_{w_x}, GR^{-1}) | i = 1, n; j = 1, m\}.$$

The weight of an edge $(s^i_{w_x} \rightarrow s^j_{w_y}, GR)$ is $A_{s^i_{w_x}|GR}^{s^j_{w_y}}$. Figure 2 shows one sense-enhanced edge.

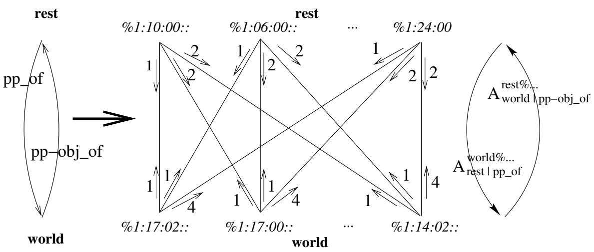


Figure 2: A sense enhanced edge, with weights induced from corpus collocations.

2.4 Word sense disambiguation

We first compute a score for each vertex (word sense) using the estimated sense preferences, traversing the dependency graph from the bottom up[4]. Each leaf is given a score of $\frac{1}{n_w}$, where n_w is the number of senses of the word w to which the leaf pertains. The score of the other vertices are the weighted sum of the scores of their grammatical dependents in the sentence under analysis:

$$Score(s^i_{w_x}) = \sum_{(w_x, w_y, GR)} \sum_{s^j_{w_y} \in S_{w_y}} A_{s^i_{w_x}|GR}^{s^j_{w_y}} \times Score(s^j_{w_y})$$

The word sense disambiguation process starts from the root node of the dependency tree. The highest ranked score for the root is chosen, and the nodes corresponding to the other senses and their edges are deleted from the graph. For each of its dependents we add the sense preferences imposed by the chosen sense to the vertex's score, and proceed with the sense selection in the same way down through the graph.

$$Score(s^j_{w_y}) = Score(s^j_{w_y}) + A_{s^i_{w_x}|GR^{-1}}^{s^j_{w_y}}$$

where (w_x, w_y, GR) $((w_y, w_x, GR^{-1}))$ is in the current sentence.

Because of data sparseness, there may be not enough evidence in the corpus to produce a clear winner, and several senses are tied. All senses are then kept, and disambiguation proceeds further. If more than one word has multiple senses left after the top-down traversal of the tree, we use two methods: random choosing from the tied senses or the sequence labeling method described in (Mihalcea, 2005). The graph's vertices are the senses that remain to be disambiguated, and its edges connect every pair of these senses (provided that they correspond to different words). The score of each vertex is initially set to 1, and the edge weights are Lesk similarity scores. The vertices are scored using a Page Rank algorithm, in which the rank at every iteration step is computed with the formula:

$$WP(a) = (1 - d) + d \sum_{b \in In(a)} \frac{w_{ba}}{\sum_{c \in Out(b)} w_{bc}} WP(b)$$

where:
a, b, c are vertices in the graph;
$WP(a)$ is the weighted PageRank score of node a;
d is the probability that there will be a jump from a given vertex to another in the graph. We use $d = 0.85$, the value set by (Brin and Page, 1998) for Google's PageRank model.
$In(a)$ is the set of a's predecessors;
$Out(a)$ is the set of a's successors.

When the vertex scores converge[5], the highest ranking vertex for each word will give the sense prediction for that word.

For multi-term expressions that are split during parsing (such as *come back*), for which there is no prediction since they do not appear as such in the parse tree, the system randomly picks one of the WordNet senses.

3 Experiments and Results

The WSD algorithm proposed is evaluated on the Senseval-2 and Senseval-3 English-all-words task test data. Table 2 shows the results obtained for fine-grained scoring. Because for each target there is a prediction, precision and recall have the same value.

[4]The up-down orientation of the graph is given by the dependency tree from which it was expanded.

[5]An aperiodic, irreducible graph is guaranteed to converge (Grimmett and Stirzaker, 1989). For every graph we built that has more than 3 nodes, the aperiodicity condition is met – it has cycles of length 2 and 3, therefore the greatest common divisor of its cycle lengths is 1. The graph is also irreducible – it has no leaves because it is highly interconnected.

POS	*Rand.*	*Seq.*	GR_{WN}	GR_{WN}^{PR}	GR_{BNC}	GR_{BNC}^{PR}
Senseval 2						
noun	41.1%	63.0%	58.9%	62.4%	54.2%	**63.3%**
verb	22.0%	31.6%	31.0%	**33.0%**	30.9%	32.7%
adjective	38.9%	**56.8%**	52.9%	**56.8%**	40.4%	**56.8%**
adverb	53.2%	57.5%	53.2%	58.8%	53.2%	**59.1%**
all	36.7%	52.1%	49.0%	52.4%	44.6%	**52.7%**
Senseval 3						
noun	42.5%	58.2%	53.2%	55.4%	40.3%	**58.6%**
verb	19.4%	40.4%	40.3%	**42.3%**	19.9%	40.0%
adjective	45.0%	56.7%	53.4%	54.5%	46.0%	**57.5%**
adverb	92.9%	92.9%	92.9%	92.9%	92.9%	92.9%
all	34.4%	50.8%	48.2%	50.1%	33.8%	**51.2%**

Table 2: Precision (= Recall) disambiguation results for Senseval English-all-words test data

Column *Random* (*Rand.*) shows a simple random baseline, and column *Sequence* (*Seq.*) shows the sequence data labelling method (Mihalcea, 2005) – one of the best performing graph-methods (Navigli and Lapata, 2007). The results presented were obtained using word similarities computed with the WordNet-Similarity-1.03 package, on a sense graph built using the marked targets in the test set. These results are not the same as those reported in (Mihalcea, 2005) for the Senseval 2 data (nouns 57.5%, verbs: 36.5%, adjective: 56.7%, adverb: 70.9%, for an average precision of 54.2%), because of the difference in computing word similarities. The other 4 columns show results obtained using grammatical relation information between words as identified by the parser. GR_{WN} includes the results obtained using the Lesk-based similarity with the syntactically-based graph and breaking ties randomly, GR_{WN}^{PR} presents results obtained in a similar configuration – only the tie breaking is done using PageRank. GR_{BNC} and GR_{BNC}^{PR} are similar with the previous two columns, only in this case the edge weights are the selectional preference scores induced from the BNC.

The performance of GR_{WN} is close to that of *Seq*. When ties are broken randomly, the computation is much faster, since we do two traversals of a small graph, while PageRank iterates until convergence (approx. 15 iterations) on graphs of average size of 1500 edges and 52 vertices (on Senseval 2 data). When PageRank is used to solve ties the performance on GR_{WN}^{PR} surpasses that of *Seq* while still being faster, having to iterate over graphs with an average of 1074 edges and 40 vertices. The computation load is not only lighter during disambiguation, but also in the data preparation stage, when similarities must be computed between every sense pair corresponding to every pair of words within a sentence (or a window of a given size).

There are other important differences. While the syntactic structure of the sentence plays no role in the *Sequence* method, it is crucial for the other methods. In the Senseval data not all words in a sentence were tagged as targets, and the *Sequence* method works only on them. This is not the case for the *GR* methods, which work with the full syntactic tree – and will disambiguate more words at a time. Also, the targets tagged in the data contain "satellites" information (e.g. *turn out*, *set up*), which may change the part of speech of the main target (e.g. *at the same time* (adv) for target *time* (noun), *out of print* (adj) for target *print* (noun)). Multi-word expressions are themselves the subject of ample research, and we could not incorporate them into our corpus-based approach. Verb particles in particular pose a problem, as most parsers will interpret the particle as a preposition or adverb. This was the case for the Senseval data, as well. On the other hand, this is a more realistic set-up, with no reliance on previously marked targets.

Selectional preferences induced from a corpus without sense annotations perform well for verbs, but overall do not perform very well by themselves. The reasons for this are multiple. The most important is data sparseness. Many sense selection preferences are 0. In order to improve this approach, we will look into more flexible methods for computing dependency pair similarities (without fixing one of the vertices as we did in this paper). Previous research in inducing sense rankings from an untagged corpus (McCarthy et al., 2004), and inducing selectional preferences at the word level (for other applications) (Erk, 2007) will provide the starting point for research in this direction.

4 Comparison with Related Work

The most similar approach to the one we describe, that has been tested on Senseval-2, is the one described in (McCarthy and Carroll, 2003). The best results reported are 51.1% precision and 23.2% recall. This implementation also used grammatical information and selectional preferences induced from a corpus to determine a disjoint partition – determined by a cut in the WordNet is-a tree – over which it computes a probability distribution conditioned by the grammatical context and a verb or adjective class.

McCarthy et al. (2004) report a disambiguation precision of 53.0% and recall of 49.0% on the Senseval-2 test data, using an approach that derives sense ranking based on word similarity and distributional analysis in a corpus.

Mihalcea (2005) reports the highest results on the Senseval-2 data obtained with a graph-based algorithm – 54.2% precision and recall. The results obtained with a PageRank algorithm applied to a sense graph built from a words within a context of a given

size are also the highest for a completely unsupervised WSD[6] system in Senseval-2.

The best result obtained by an unsupervised system on the Senseval-3 data is reported by Strapparava et al. (2004) – 58.3%. This implementation uses WordNet-Domains, a version of WordNet enhanced with domain information (e.g. economy, geography). The domain of a given text is automatically detected, and this information will constrain the possible senses of words in the given text.

For Senseval 3 data, using a graph method with the *Key Player Problem* to measure vertex relevance, Navigli and Lapata (2007) report very close results to (Strapparava et al., 2004) on nouns and adjectives, and lower scores for verbs (F1-scores: 61.9% for nouns, 62.8% for adjectives, 36.1% for verbs compared with 62.2% for nouns, 66.9% for adjectives, 50.4% for verbs). Mihalcea (2005) reports an overall score of 52.2% for this data.

It is interesting to look at the dependency tree we used for WSD from the point of view of graph connectivity measures (Navigli and Lapata, 2007). To determine the importance of a node in a graph, whether it represents the words and their senses in a given context, or people in a social network, one can use different measures. According to grammatical theories, the importance of a node in the sentence parse tree is given by the phrase type it heads, and the number of words it thus dominates. From this point of view, the top-down propagation of senses traverses and disambiguates the tree in order of the decreasing importance of nodes. Other methods could be used as well, such as disambiguating first the most highly connected nodes – the ones with the most sense constraints.

5 Conclusions

We have studied the impact of grammatical information for constraining and guiding the word sense disambiguation process in an unsupervised all-words setup. Compared with graph methods, the approach we described is computationally lighter, while performing at the same level on Senseval-2 and Senseval-3 all-words tasks test data. Grammatical constraints serve both to limit the number of word-senses pair similarities necessary, and also to estimate selectional preferences from an untagged corpus.

Using only grammatically motivated connections leads to better disambiguation of verbs for both Senseval-2 and Senseval-3 test data, but while the difference is consistent (1.4%, 1.9%) it is not statistically significant.

We explored a new method for estimating sense association strength from a sense-untagged corpus. Disambiguation when using sense relatedness computed from WordNet is very close in performance with disambiguation based on sense association strength computed from the British National Corpus, and on a par with state-of-the-art unsupervised systems on Senseval-2. This indicates that grammatical relations and automatically derived sense association preference scores from a corpus have high potential for unsupervised all-word sense disambiguation.

[6] As opposed to other unsupervised approaches, the sense frequency information from WordNet was not used.

References

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlap as a measure of semantic relatedness. In *Proc. of IJCAI-03,* Acapulco, Mexico, 9–15 August, 2003, pages 805–810.

Sergey Brin and Larry Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:1–7.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. of ACL-07*, pages 33–40.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Computational Linguistics*, 28(2):187–206.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure. In *Proc. of LREC-06*.

Michael Elhadad, Kathleen R. McKeown, and Jaques Robin. 1997. Floating constraints in lexical choice. *Computational Linguistics*, 23(2):195–239.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proc. of ACL-07*, pages 216–223.

Geoffrey Grimmett and David Stirzaker. 1989. *Probability and Random Processes*. Oxford University Press.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The Sketch Engine. In *Proc. of EuraLex-04*, pages 105–116.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL-03*, pages 423–430.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics*, 24(2):217–244.

Diana McCarthy and John Carroll. 2003. Disambiguating nouns, verbs and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proc. of ACL-04,* Barcelona, Spain, 21–26 July 2004, pages 280–287.

Rada Mihalcea. 2005. Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proc. of HLT-EMNLP-05*, pages 411–418.

Roberto Navigli and Mirella Lapata. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proc. of IJCAI-07*, pages 1683–1688.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: all-words and verb lexical sample. In *Proc. of the ACL SENSEVAL-2 Workshop, Toulouse, France, 2001*.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity – Measuring the relatedness of concepts. In *Proc. of HLT-NAACL-04,* Boston, Mass., 2–7 May, 2004, pages 267–270.

Philip Resnik. 1996. Selectional constraints: an information-theoretic model and its computational realization. *Cognition*, (61):127–159, November.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proc. of the ACL SENSEVAL-3 Workshop, Barcelona, Spain, 2004*, pages 41–43.

Jiri Stetina, Sadao Kurohashi, and Makoto Nagao. 1998. General word sense disambiguation method based on a full sentential context. In Sanda Harabagiu, editor, *PROC. of Use of WordNet in Natural Language Processing Systems*, pages 1–8.

Carlo Strapparava, Alfio Gliozzo, and Claudio Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation. In *Proc. of the ACL SENSEVAL-3 Workshop, Barcelona, Spain, 2004*, pages 229–234.

Michael Sussna. 1993. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proc. of the CIKM-93*, pages 67–74.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivalling supervised methods. In *Proc. of ACL-05,* Cambridge, Mass., 26–30 June 1995, pages 189–196.

Syntactic and Semantic Frames in PrepNet

Patrick Saint-Dizier
IRIT-CNRS
118, route de Narbonne
31062 Toulouse cedex France
stdizier@irit.fr

Abstract

In this paper, we present an in depth revision of the preliminary version of PrepNet, an lexical semantics database of preposition behaviors. This revision includes a much more detailed structure for the language realization level.

1 Aims

Describing the syntax and the semantics of prepositions, in a way similar to verbs (e.g. in FrameNet (www.icsi.berkeley.edu/ framenet/), or VerbNet (www.cis.upenn.edu/verbnet/)) or to nouns (as in WordNet and EuroWordNet) is obviously a very challenging, but necessary task. Prepositions turn out to be a very useful category in a number of applications such as indexing, knowledge extraction, textual entailment and question answering since they convey basic meanings of much interest like instruments, means, comparisons, amounts, approximations, localizations, etc. They must necessarily be taken into account—and rendered accurately—for effective machine translation and lexical choice in language generation. However, so far, prepositions have not been studied in NLP circles as extensively as nouns and verbs.

PrepNet (http://www.irit.fr/recherches/ILPL/Site-Equipe/ILPL.htm under revision) is a framework that aims at constructing a repository of preposition syntactic and semantic behaviors. PrepNet is structured in two levels:

- the abstract notion level: global, language independent, characterization of preposition senses in an abstract way, where frames represent some generic semantic aspects of these notions,
- the language realization levels that deal with realizations for various languages, using a variety of marks (postpositions, affixes, compounds, etc.). However, we will keep the term 'preposition' hereafter for all these marks.

We present here a revised version of a preliminary PrepNet. Besides some simplifications of the structure, innovations mainly concern the structures provided at the language realization level, motivated by the study of a few notions for a variety of languages. We provide the means to describe syntactic subcategorization forms, as well as semantic and pragmatic restrictions on uses. It seems that our view is now usable in a number of languages, but some are still resistant, like, e.g. Malay.

2 Related work

There is quite a lot of literature on prepositions in psycholinguistics circles, and some in AI and in cognitive sciences (Horno Cheliz, 82), (Cervioni, 91), (Lindstomberg 97), (Mari, 00), (Pesetzky, 82), (Talmy 76, 85), but less in NLP (Saint-Dizier, 06).

A quite old, but still of interest work (Spark-Jones et al. 85) proposes, via cases or roles, a structure for prepositions, and their relations to verbs.

The basis and the starting point of our research was developed about 10 years ago by Bonnie Dorr, it is accessible at:
www.umiacs.umd.edu/b̃onnie/
AZ-preps-English.lcs.
This is a very large database of preposition semantic

representations, characterized by their LCS representation and, sometimes, by a thematic grid. It was concieved for machine translation tasks, which explains some of its features. There are about 500 entries (compared to our 165 abstract notions), for probably all English prepositions. Finally, the Preposition Project offers a lexicographic view of prepositions in English.

3 Main features of PrepNet

Within the PrepNet framework, we have identified so far 195 preposition senses (which may have subsenses in particular languages), which can be represented in the Lexical Conceptual Structure framework on the basis of 65 primitives, based on English preposition names (on, near, with, etc.). These senses reflect the variety of primitive notions conveyed by prepositions. Abstract notion representations may be a composition of several primitives. Primitives are viewed here as linguistic macros, which can then be interpreted depending on the environment (e.g. Euclidean geometry for spatial prepositions, fuzzy logic axioms for the notion of approximation).

To elaborate an adequate formalism for the syntactic and semantic aspects of prepositions we want to encode in PrepNet, we have studied in depth preposition realizations in language around the abstract notions of theme and approximation (French, Spanish, Catalan, English, Thai) and instruments (for German, Italian, Spanish, French, Arabic and Berber, Thai, Bahasa Malaysia, Hindi, Urdu, Kashmiri, Bengali, and Filipino). This latter notion is much wider than the first two, and has a large variety of realization parameters, which greatly contributed to this second version of PrepNet. A multilingual analysis needs to be somewhat transcategorial and both syntactic and semantic.

4 Preposition usage characterizations using Frames

4.1 General overview of abstract notions

In PrepNet, preposition senses are characterized by means of abstract notions which capture senses in a conceptual and language independent way. These abstract notions have been defined from corpus analysis and bilingual dictionaries (in particular the German-French Harrap's, which has an excellent conceptual approach to translation). From corpora, focussing on French, Spanish and English, we have studied every preposition lexeme, its distributions and constraints, and we have identified manually the different meanings they convey. We have then formed groups of senses out of which the abstract notions emerged. This took about 1.5 man/year of work, preposition being not as numerous as e.g. verbs or adjectives (about 50 to 100).

The first level, the abstract notions, is organized as follows:
- a first level characterizes a **semantic family**, of a level roughly comparable to thematic roles: localization, manner, quantity, accompaniment, etc.,
- a second level accounts for the different **facets** of a semantic family, e.g. source, destination, via, and fixed position for the localization family,
- a third level characterizes, roughly speaking, the **modalities of a facet** when appropriate. For example, the facet *manner and attitudes* is decomposed into 3 modalities: *basic manner, manner by comparison and manner with a reference point*. Due to space limitations, this latter level will not be much developed here.

Abstract notions are the following (revised from (Saint-Dizier, 05)):

- **Localization** with facets:
  - source, - destination, - via/passage, - fixed position.
  From an ontological point of view, all of theses facets can, a priori, apply to spatial, temporal or to more abstract arguments.

- **Quantity** with facets:
  - numerical or referential quantity, - frequency and iterativity, - proportion or ratio.

- **Manner** with facets:
  - manners and attitudes, - means (instrument or abstract), - imitation, agreement or analogy.
  Imitation: *he walks like a robot; agreement: he behaves according to the law,*

- **Accompaniment** with facets:
  - adjunction, - simultaneity of events (co-events), - inclusion, - exclusion.

Adjunction : *flat with terrace / steak with French fries / tea with milk*, Exclusion: *they all came except Paul*.

- **Choice and exchange** with facets:
- exchange, - choice or alternative, - substitution.
Substitution : *sign for your child*, Choice: *among all my friends, he is the funniest one*.

- **Causality** with facets :
- cause, - goal or consequence, - intention - purpose.
Cause: *the rock fell under the action of frost*.

- **Opposition** with two ontological distinctions: physical opposition and psychological or epistemic opposition, e.g.: *to act contrary to one's interests*.

- **Ordering** with facets:
- priority, - subordination, - hierarchy, - ranking, - degree of importance.
Ranking : *at school, she is ahead of me*.

- **Instrument** (see below),

- Other groups: **- Theme, - in spite of, - comparison**.
Theme: *a book concerning dinosaurs*.

4.2 Representation of abstract notions

An abstract notion is characterized by:

1. **a name and a gloss**, that informally describe the abstract notion at stake,

2. a **conceptual representation**, in simplified LCS form,

3. **inferential patterns** and presuppositions, which will not be developed here.

4.3 Representation of the language level

While we have a unique, language independent, structure for abstract notions, we have a set of descriptions for each language. At this level, we may also have semantic subdivisions whenever relevant, called strata. Let us study here the direct usages, i.e. those which are not a priori metaphorical or in any other form of meaning shift. Our approach, however, integrates the possibility to describe these shifts either directly or via rules.

The language level descriptions include at the moment the following features:

- syntactic frames: the syntactic subcategorization frames the preposition heads (possibly in conjunction with other predicates like a verb), with some statistics on usage frequency. Frames are a little bit complex since they need to take into account (1) elements not completely headed by the preposition but which nevertheless play a major role (the verb and the 'external argument' of the preposition) and (2) the structure the preposition heads (in general an NP, possibly an S). This is realized by corpus inspection. In addition, alternations prepositions may undergo are given and some grammatical movements (such as fronting).

- semantic and domain restrictions: each argument in the frame may have selectional restrictions. These allow us to identify different language realizations. Restrictions may also be related to domains and not to arguments.

- pragmatic aspects: prepositions convey a number of pragmatic factors such as: stress (identifying a new focus), polarity, illocutionary force, formal character, etc. These are mainly given to restrict usages.

4.4 Basic case: the VIA notion

The facet VIA of the 'localisation' family describes a movement via a passage. The abstract notion frame is defined as follows:

```
VIA
'An entity X moving via a location Y'
representation: X : via(loc, Y)
```

This frame reads as follows: after the name and the gloss of the notion, we find a simple, LCS-based, semantic representation where X is the external argument, and loc specifies the domain: localization (other cases are assumed to be derived by metaphor).

Let us now consider the language level. In French, the by-default associated synset is {*par*, *via*} . X and Y are restricted respectively to concrete entities and location, the verb is restricted to inherently directed motion (in B. Levin's terminology), as in:

passer par la porte, *transiter par la Belgique*, *Paris-Strasbourg via Nancy.*
Syntactic frames examples are (given informally, in readable format):
`[X(np,subj), Verb, Y(np,obj,optional), preposition, Z(np,obj2)],`
[Y(compound NP, +loc), preposition, Z(np, +loc)], etc.

Next, we also have a specific case (a strata) where the passage is narrow. In that case, the synset is {*à travers, au travers de, dans*}, and Verbs are either inherently directed motion or perception verbs (*regarder dans le téléscope, regarder à travers la grille*).

4.5 Compound forms: via under

The abstract notion VIA has a few strata that correspond to compound notions that express a more precise trajectory like via under, via above. For example, in French, to express this notion, the preposition *par* is combined with a fixed location preposition such as *dessous*, *dessus* etc. to form compounds such as: *par dessus*, *par dessous* (via under, via above). The frame structure remains the same, except that the semantic representation has then an embedded functional structure:

```
VIA UNDER
'An entity moving via under a location'
representation: X : via(loc, under(loc,Y))
```

At the language realization level, the French synset is: {par dessous }.

5 A more complex case: dealing with instruments

The study of the abstract notion of instrumentality, as reported in (Kawtrakul et alii, 06), has led us to revise and largely improve the language level formalism. Let us report here some of its main facets. In this work, 12 languages from 5 linguistic families are studied: Thai, Malay, Hindi, Urdu, Kashmiri, Bengali, German, Spanish, French, Italian, Arabic and Berber. Filipino has been recently considered.

Besides basic syntactic frames, of much interest is that most of these languages use other forms than 'prepositions' to realize these abstract notions: postpositions, various kinds of affixes, verb compounds, etc. A number of the languages studied have an instrumental case.

Our investigations tend to show that we can have on the one hand a stable abstract frame that represents the semantic and some pragmatic aspects of the abstract notion and, on the other hand, at the language realization level, a description of the behaviors of 'prepositions' in the various languages. We do not establish any direct connection between two preposition realizations in two languages. The relation, in terms of translation, is established via the set of restrictions imposed on each lexicalization that corresponds the best to the restrictions imposed on the argument Y. The impact of the other elements (arguments and verb) remains to be explored. Each language has an independent description.

5.1 Representing the abstract notion

The generic frame for instruments is as follows:

```
INSTRUMENT
'An actor X uses an object Z (the instrument)
to reach a goal E'
X, Y : by_means_of(E, Z)
```

In this frame, an event E is introduced to refer to the goal, e. g. 'cut bread' as in *John cuts bread with a knife*. Note also that in the semantic representation, the instrumental expression has wider scope over the event: this reflects the fact that most adjuncts have scope over propositions (predicate and its arguments).

In terms of restrictions, a major difficulty is the prototypicality of instruments. At a conceptual level, it is quite difficult to characterize what is a prototypical instrument for a given action. Each event has its own prototypical instrument, making corpus studies extremely large, probably unfeasable. For the time being, we leave the type of the instrument largely open. However, at the language realization level, we may have some useful restrictions, as will be seen below.

5.2 Dealing with language realizations

Let us now present a variety of language realizations that motivated the different facets of the formalism we have developed. In each case, we have a by-default synset of marks, and more restricted sets of marks for specific cases, related in particular to the semantic type of the instrument, but also to pragmatic effects or domains of discourse.

The different language variations presented below have been elaborated in several steps via cor-

pus and dictionaries. A first set of utterances was collected by using the various prepositions in each language and by analyzing the usage restrictions observed. Then we constructed a second set of utterances to confirm the analysis, by attempting to find counter examples. Counter examples always exist, but they must remain marginal for the analysis to be confirmed. Analysis was done independently for each language in order to avoid any influence. Much more data can be found in (Kawtrakul et alii, 06).

Let us now present language realization levels for a few quite diverse languages.

French by-default synset: `[avec, par, au moyen de, grâce à, à l'aide de, à travers]`.
some syntactic structures:
`[X(subj) Verb(action) Y(obj) preposition Z(np or S)], ['utiliser' Z 'pour' Verb(action, infinitive) Y]`, etc..
More informally, other syntactic properties are: the instrument is in general an adjunct to the VP, it therefore has the properties of such types of adjuncts. It undergoes the alternation: 'Characteristic property of instrument' (Levin 86) and a few other movements, e.g.: fronting. Finally, it cannot be inserted between the verb and the object.
Usage restrictions: introduces a focus on the instrument (in particular to focus on non prototypical instruments): `au moyen de`.
polarity: positive: `grâce à`

German by-default synset: `[mit, mit Hilfe von, mittels, durch, anhand, kraft, dank, per.]`
Of interest here are the domain and pragmatic restrictions on preposition usages, e.g.:
domain: juridical, psychological: `kraft, anhand`
formal usage: `mittels`
focus: `mittels, mit Hilfe von`
instrumental manner: `durch`.

Hindi by-default synset: `[se, me, ke karaan, ke dwaraa, kar, karaan, dwara]`
the syntactic frame encodes here postpositions, case marks and the SOV form: `[X(subject, ergative), Y(object, accusative), Z(adjunct), postposition, Verb(action)]`.
This form is a priori very regular, Z is an NP or an S. Y and Z can occasionally be permuted. Let us note some interesting usage restrictions:
instrument type: concrete: `se`
instrument type: abstract: `dwAra`
instrument type: means of transportation: `me`
involvement of instrument: agentive: `ke dwara`, causal `ke karaan`
instrumental path: `me, se`.

Concerning other Northern India languages, Urdu has about the same distribution and distinctions, while Kashmiri and most notably Bengali have some more distinctions, with the use of a large number of prefixes and suffixes, which are expressed in the subcat frame by means of features. Thai is relatively straightforward, prepositions may be even omitted.

Filipino is close to Malay and Indonesian in terms of structure, except that it is basically a VSO language. It has also a large number of marks to capture the notion of preposition: `[ng, kay, kina, sa]`, as in:
Pinalo niya ang aso ng patpat (litt. hits he the dog with a stick).
The syntaxtic structure is therefore: `[Verb, X(subj), Y(obj), preposition, Z]`.

So far, the formalism we have elaborated allows us to encode syntactic frames, restrictions, case marks, prefixes and suffixes as well as postpositions. However, languages of the malayo-polynesian family raise additional problems which are not so easy to capture. Let us just, for the sake of illustration, survey a few aspects here for Malay and Filipino.

Malay has three ways to introduce instruments: preposition + NP, affixes and compounding. Affixed words are built from stems which are instrumental nouns, this allows for the construction of the equivalent of PPs, based on the prototypical use of the instrumental noun. The most common being: prefixes: beR- (e.g. from kuda, horse, berkuda, on horseback), meN- (e.g. from kunci, key, mengunci, lock with key), prefix + suffix: meN- + -kan (e.g. from paku, nail, memakukan, to fasten with nails), and with suffix -i (e.g. from ubat, medicine, mengubati, by means of medicine). At the moment, we feel it may be confusing to add derivational morphology considerations (with many restrictions) into syntactic frames, probably an additional means would be necessary.

Similarly, Filipino has also a large number of marks that play the role of prepositions, one of which is viewed as an anteposed particle, working as a determiner.

6 Perspectives

Although we have stabilized semantic notions and some formalisms for the representation of preposition behaviors over a number of languages, PrepNet is still in a development stage. The descriptions over various languages are huge tasks. Our method for the future will be to proceed by notion and study a variety of languages to have a better grasp at the semantic distinctions and language realizations, as we did for instruments. For each case, a dedicated method is often required. Descriptions are encoded in XML, so that data can be easily shared.

Although the study of prepositions is an interesting topic in itself, it is of much interest to investigate how this work can be integrated into larger frameworks such as FrameNet or VerbNet, and this is one of our major prospective.

FrameNet says little about prepositions, but it has a few frames such as Accompaniment which are of interest. The roles defined in FrameNet are more accurate than the abstract notions of PrepNet, which aims at a relatively generic description. Those roles are related to a variety of situations which are not necessarily introduced by prepositions. However, a preliminary, exploratory, task could be to attempt to classify FrameNet roles under the main abstract notions of PrepNet.

VerbNet uses a quite detailed list of thematic roles which have some similarities with the top nodes of PrepNet abstract notions hiereracy. In a VerbNet frame, the syntax slot could be enriched by preposition type (abstract notions) restrictions. Similarly, predefined primitives such as location or direction are really close to our semantic representations in LCS, but they are used in a different manner. PrepNet has additional primitives to handle argument and non argument structures (e.g. approximation).

References

Cannesson, E., Saint-Dizier, P. (2001), *A general framework for the representation of prepositions in French*, ACL01 WSD workshop, Philadelphia.

Carmen Horno Chéliz, M. del, (2002), *Lo que la preposición esconde*, University of Zaragoza press.

Cervioni, J., (1991), *La préposition: Etude sémantique et pragmatique*, Duculot, Paris.

Dorr, B., Olsen, M.B., (1997), *Deriving Verbal and Compositional Verbal Aspect for NLP Applications*, proc. ACL'97, Madrid.

Dorr, B. J., Garman, J., and Weinberg, A., (1995), *From Syntactic Encodings to Thematic Roles: Building Lexical Entries for Interlingual MT*, Machine Translation, 9:3-4, pp.71-100.

Fellbaum, C., (1993), *English Verbs as Semantic Net*, Journal of Lexicography, vol. 6, Oxford University Press.

Jackendoff, R., (1990), *Semantic Structures*, MIT Press.

Kawtrakul, A. et alii (2006), A Multilingual Analysis of the Notion of Instrumentality, proc. EACL workshop on prepositions, Trento.

Levin, B., (1993), *Verb Semantic Classes: a Preliminary Investigation*, Chicago University Press.

Lindstromberg, S. (1997), *English Prepositions Explained*, John Benjamins.

Saint-Dizier, P., (2005), *PrepNet: a Framework for Describing Prepositions: Preliminary Investigation Results*, IWCS05, Tilburg.

Spark-Jones, K., Boguraev, B., A note on a study of cases, research note, dec. 85.

Talmy, L. (1976), *Semantic Causative Types*, In M. Shibatani (ed.), Syntax and Semantics 6: The Grammar of Causative Constructions. New York: Academic Press, pp. 43-116.

Talmy, L., (1985), *Lexicalization Patterns: Semantic Structure in Lexical Forms*, in *Language Typology and Syntactic Description 3: Grammatical Categories and the Lexicon*, T. Shopen (ed.), 57-149, Cambridge University Press.

Villavicencio, A., (2006) *Verb-particle Constructions in the WWW*, in P. Saint-Dizier (ed), *Syntax and Semantics of Prepositions*, Kluwer Academic.

Wierzbicka, A. (1992), *Semantic Primitives and Semantic Fields*, in A. Lehrer and E.F. Kittay (eds.), Frames, Fields and Contrasts. Hillsdale: Lawrence Erlbaum Associates, pp. 208-227.

Automatic Classification of English Verbs Using Rich Syntactic Features

Lin Sun and Anna Korhonen
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
ls418,alk23@cl.cam.ac.uk

Yuval Krymolowski
Department of Computer Science
University of Haifa
31905, Haifa
Israel
yuvalkry@gmail.com

Abstract

Previous research has shown that syntactic features are the most informative features in automatic verb classification. We experiment with a new, rich feature set, extracted from a large automatically acquired subcategorisation lexicon for English, which incorporates information about arguments as well as adjuncts. We evaluate this feature set using a set of supervised classifiers, most of which are new to the task. The best classifier (based on Maximum Entropy) yields the promising accuracy of 60.1% in classifying 204 verbs to 17 Levin (1993) classes. We discuss the impact of this result on the state-of-art, and propose avenues for future work.

1 Introduction

Recent research shows that it is possible, using current natural language processing (NLP) and machine learning technology, to automatically induce lexical classes from corpus data with promising accuracy (Merlo and Stevenson, 2001; Korhonen et al., 2003; Schulte im Walde, 2006; Joanis et al., 2007). This research is interesting, since lexical classifications, when tailored to the application and domain in question, can provide an effective means to deal with a number of important NLP tasks (e.g. parsing, word sense disambiguation, semantic role labeling), as well as enhance performance in many applications (e.g. information extraction, question-answering, machine translation) (Dorr, 1997; Prescher et al., 2000; Swier and Stevenson, 2004; Dang, 2004; Shi and Mihalcea, 2005).

Lexical classes are useful because they capture generalizations over a range of (cross-)linguistic properties. Being defined in terms of similar meaning components and (morpho-)syntactic behaviour of words (Jackendoff, 1990; Levin, 1993) they generally incorporate a wider range of properties than e.g. classes defined solely on semantic grounds (Miller, 1990). They can be used to build a lexical organization which effectively captures generalizations and predicts much of the syntax and semantics of a new word by associating it with an appropriate class. This can help compensate for lack of data for individual words in NLP.

Large-scale exploitation of lexical classes in real-world or domain-sensitive tasks has not been possible because existing manually built classifications are incomprehensive. They are expensive to extend and do not incorporate important statistical information about the likelihood of different classes for words. Automatic classification is a better alternative. It is cost-effective and gathers statistical information as a side-effect of the acquisition process.

Most work on automatic classification has focussed on verbs which are typically the main predicates in sentences. Syntactic features have proved the most informative in verb classification. Experiments have been reported using both (i) deep syntactic features (e.g. subcategorization frames (SCFs)) extracted using parsers and subcategorisation acquisition systems (Schulte im Walde, 2000; Korhonen et al., 2003; Schulte im Walde, 2006) and (ii) shallow ones (e.g. NPs/PPs preceding/following verbs) extracted using taggers and chunkers (Merlo and Stevenson, 2001; Joanis et al., 2007).

(i) correspond closely with features used for manual classification (Levin, 1993). They have proved successful in the classification of German (Schulte im Walde, 2006) and English verbs (Korhonen et al., 2003). Yet promising results have also been reported when using (ii) for English verb classification (Merlo and Stevenson, 2001; Joanis et al., 2007). This may indicate that (i) are optimal for the task when combined with additional syntactic information from (ii).

We investigate this matter by experimenting with a new, rich feature set which incorporates information about SCFs (arguments) as well as adjuncts. It was extracted from VALEX, a large automatically acquired SCF lexicon for English (Korhonen et al., 2006). We evaluate the feature set thoroughly using set of supervised classifiers, most of which are new in verb classification. The best performing classifier (Maximum Entropy) yields the accuracy of 60.1% on classifying 204 verbs into 17 Levin (1993) classes. This result is good, considering that we performed no sophisticated feature engineering or selection based on the properties of the target classification (Joanis et al., 2007). We propose various avenues for future work.

We introduce our target classification in section 2 and syntactic features in section 3. The classification techniques are presented in section 4. Details of the experimental evaluation are supplied in section 5. Section 6 provides discussion and concludes with directions for future work.

2 Test Verbs and Classes

We adopt as a target classification Levin's (1993) well-known taxonomy where verbs taking similar diathesis alternations are assumed to share meaning components and are organized into a semantically coherent class. For instance, the class of "*Break* Verbs" (class 45.1) is partially characterized by its participation in the following alternations:

1. **Causative/inchoative alternation**:
Tony broke the window ↔ *The window broke*

2. **Middle alternation**:
Tony broke the window ↔ *The window broke easily*

3. **Instrument subject alternation**:
Tony broke the window with the hammer ↔ *The hammer broke the window*

	LEVIN CLASS	EXAMPLE VERBS
9.1	PUT	bury, place, install, mount, put
10.1	REMOVE	remove, abolish, eject, extract, deduct
11.1	SEND	ship, post, send, mail, transmit
13.5.1	GET	win, gain, earn, buy, get
18.1	HIT	beat, slap, bang, knock, pound
22.2	AMALGAMATE	contrast, match, overlap, unite, unify
29.2	CHARACTERIZE	envisage, portray, regard, treat, enlist
30.3	PEER	listen, stare, look, glance, gaze
31.1	AMUSE	delight, scare, shock, confuse, upset
36.1	CORRESPOND	cooperate, collide, concur, mate, flirt
37.3	MANNER OF SPEAKING	shout, yell, moan, mutter, murmur
37.7	SAY	say, reply, mention, state, report
40.2	NONVERBAL EXPRESSION	smile, laugh, grin, sigh, gas
43.1	LIGHT EMISSION	shine, flash, flare, glow, blaze
45.4	CHANGE OF STATE	soften, weaken, melt, narrow, deepen
47.3	MODES OF BEING WITH MOTION	quake, falter, sway, swirl, teeter
51.3.2	RUN	swim, fly, walk, slide, run

Table 1: Test classes and example verbs

Alternations are expressed as pairs of SCFs. Additional properties related to syntax, morphology and extended meanings of member verbs are specified with some classes. The taxonomy provides a classification of 4,186 verb senses into 48 broad and 192 fine-grained classes according to their participation in 79 alternations involving NP and PP complements.

We selected 17 fine-grained classes and 12 member verbs per class (table 2) for experimentation. The small test set enabled us to evaluate our results thoroughly. The classes were selected to (i) include both syntactically and semantically similar and different classes (to vary the difficulty of the classification task), and to (ii) have enough member verbs whose predominant sense belongs to the class in question (we verified this according to the method described in (Korhonen et al., 2006)). As VALEX was designed to maximise coverage most test verbs had 1000-9000 occurrences in the lexicon.

3 Syntactic Features

We employed as features distributions of SCFs specific to given verbs. We extracted them from the recent VALEX (Korhonen et al., 2006) lexicon which provides SCF frequency information for 6,397 English verbs. VALEX was acquired automatically from five large corpora and the Web (using up to 10,000 occurrences per verb) using the subcategorization acquisition system of Briscoe and Carroll (1997). The system incorporates RASP, a domain-independent robust statistical parser (Briscoe and

Carroll, 2002), and a SCF classifier which identifies 163 verbal SCFs. The basic SCFs abstract over lexically-governed particles and prepositions and predicate selectional preferences.

We used the noisy *unfiltered* version of VALEX which includes 33 SCFs per verb on average[1]. Some are genuine SCFs but some express adjuncts (e.g. *I sang in the party* could be SCF PP). A lexical entry for each verb and SCF combination provides e.g. the frequency of the entry (in active and passive) in corpora, the POS tags of verb tokens, the argument heads in argument positions, and the prepositions in PP slots. We experimented with three feature sets:

1. **Feature set 1**: SCFs and their frequencies
2. **Feature set 2**: Feature set 1 with two high frequency PP frames parameterized for prepositions: the simple PP (e.g. *they apologized* **to** *him*) and NP-PP (e.g. *he removed the shoes* **from** *the bag*) frames.
3. **Feature set 3**: Feature set 2 with three additional high frequency PP frames parameterized for prepositions: the NP-FOR-NP (e.g. *he bought a book* **for** *him*), NP-TO-NP (e.g. *he gave a kiss* **to** *her*), and OC-AP, EQUI, AS (e.g. *he condemned him* **as** *stupid*) frames.

In feature sets 2 and 3, 2-5 PP SCFs were refined according to the prepositions provided in the VALEX SCF entries (e.g. PP_at, PP_on, PP_in) because Levin specifies prepositions with some SCFs / classes. The scope was restricted to the 3-5 highest ranked PP SCFs to reduce the effects of sparse data.

4 Classification

4.1 Preparing the Data

A feature vector was constructed for each verb. VALEX includes 107, 287 and 305 SCF types for feature sets 1, 2, and 3, respectively. Each feature corresponds to a SCF type, and its value is the relative frequency of the SCF with the verb in question. Some of the feature values are zero, because most verbs take only a subset of the possible SCFs.

4.2 Machine Learning Methods

We implemented three methods for classification: the K nearest neighbours (KNN), support vector machines (SVM), and maximum entropy (ME). To our knowledge, only SVM has been previously used for verb classification. The free parameters were optimised for each feature set by (i) defining the value range (as explained below), and by (ii) searching for the optimal value on the training data using 10 fold cross validation (section 5.2).

4.2.1 K Nearest Neighbours

KNN is a memory-based classification method based on the distances between verbs in the feature space. For each verb in the test data, we measure its distance to each verb in the training data. The verb class label is the most frequent label in the top K closest training verbs. We use the entropy-based Jensen-Shannon (JS) divergence as the distance measure:

$$JS(P,Q) = \frac{1}{2}\left[D(P\|\tfrac{P+Q}{2}) + D(Q\|\tfrac{P+Q}{2})\right]$$

The range of the parameter K is 2-20.

4.2.2 Support Vector Machines

SVM (Vapnik, 1995) tries to find a maximal margin hyperplane to separate between two groups of verb feature vectors. In practice, a linear hyperplane does not always exist. SVM uses a kernel function to map the original feature vectors to higher dimension space. The 'maximal margin' optimizes our choice of dimensionality to avoid over-fitting. We use Chang and Lin (2001) 's LIBSVM library to implement the SVM. Following Hsu et al. (2003), we use the radial basis function as the kernel function:

$$K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2), \gamma > 0$$

γ and the cost of the error term C (the penalty for margin errors) are optimized. The search ranges of Hsu et al. (2003) are used:

$$C = 2^{-5}, 2^{-3}, \ldots, 2^{15}, 2^{17}\ ; \gamma = 2^{-17}, 2^{-15}, \ldots, 2^{1}, 2^{3}$$

4.2.3 Maximum Entropy

ME constructs a probabilistic model that maximizes entropy on test data subject to a set of feature constraints. If verb x is in class 10.1 and takes the SCF 49 (NP-PP) with the relative frequency of 0.6 in feature function f, we have

$$f(x,y) = 0.6 \text{ if } y = 10.1 \text{ and } x = 49$$

The expected value of a feature f with respect to the empirical distribution (training data) is

$$\tilde{\mathcal{E}}(f) \equiv \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

The expected value of the feature f (on test data) with respect to the model $p(y|x)$ is

[1]The SCF accuracy of this lexicon is 23.7 F-measure, see (Korhonen et al., 2006) for details.

$$\mathcal{E}(f) \equiv \sum_{x,y} \tilde{p}(x)p(y|x)f(x,y)$$

$\tilde{p}(x)$ is the empirical distribution of x in the training data. We constrain $\mathcal{E}(f)$ to be the same as $\tilde{\mathcal{E}}(f)$

$$\mathcal{E}(f) = \tilde{\mathcal{E}}(f)$$

The model must maximize the entropy $H(Y|X)$

$$H(Y|X) \equiv -\sum_{x,y} \tilde{p}(x)p(y|x)\log p(y|x)$$

The constraint-optimization problem is solved by the Lagrange multiplier (Pietra et al., 1997). We used Zhang (2004)'s maximum entropy toolkit for implementation. The number of iterations i (5-50) of the parameter estimation algorithm is optimised.

5 Experiments

5.1 Methodology

We split the data into training and test sets using two methods. The first is 'leave one out' *cross-validation* where one verb in each class is held out as test data, and the remaining N-1 (i.e. 11) verbs are used as training data. The overall accuracy is the average accuracy of N rounds. The second method is *resampling*. For each class, 3 verbs are selected randomly as test data, and 9 are used as training data. The process is repeated 30 times, and the average result is recorded.

5.2 Measures

The methods are evaluated using first accuracy – the percentage of correct classifications out of all the classifications:

$$Accuracy = \frac{truePositives}{truePositives + falseNegatives}$$

When evaluating the performance at class level, precision and recall are calculated as follows:

$$Precision = \frac{truePositives}{truePositives + falsePositives}$$

$$Recall = \frac{truePositives}{truePositives + falseNegatives}$$

F-score is the balance over recall and precision. We report the average F-score over the 17 classes. Given there are 17 classes in the data, the accuracy of randomly assigning a verb into one of the 17 classes is $1/17 \approx 5.8\%$.

5.3 Results from Quantitative Evaluation

Table 2 shows the average performance of each classifier and feature set according to 'leave one out' cross-validation[2]. Each classifier performs considerably better than the random baseline. The simple KNN method produces the lowest accuracy (44.1-54.9) and SVM and ME the best (47.1-57.9 and 47.5-59.3, respectively).

The performance of all methods improves sharply when moving from the feature set 1 to the refined feature set 2: both accuracy and F-measure improve by over 10%. When moving from feature set 2 to the sparser feature set 3 (which includes a higher number of low frequency PP features) KNN worsens clearly (c. 5% in accuracy and F-measure) while the improvement in other methods is very small. This suggests that KNN deals worse than other methods with sparse data.

The resampling results in table 3 reveal that some classifiers perform worse than others when less training data is available[3]. KNN produces considerably lower results, particularly with the sparse feature set 3: 28.2 F-measure vs. 48.2 with cross-validation. Also SVM performs worse with feature set 3: 54.6 F-measure vs. 58.2 with cross-validation. ME thus appears the most robust method with smaller training data, producing results comparable with those in cross-validation.

Figure 1 shows the F-measure for 17 individual classes when the methods are used with feature set 3. Levin classes 40.2, 29.2, and 37.3 (see table 2) (the ones taking fewer prepositions with higher frequency) have the best average performance (65% or more), and classes 47.3, 45.4 and 18.1 the worst (40% or less). ME outperforms SVM with 9 of the 17 classes.

5.4 Qualitative Evaluation

We did some qualitative analysis to trace the origin of error types produced by ME with feature set 3. Examination of the worst performing class 47.3 (MODES OF BEING INVOLVING MOTION verbs) illustrates well the various error types. 10 of the 12 verbs in this class are classified incorrectly:

- **3** in class 43.1 (LIGHT EMISSION verbs): Verbs in 47.3 and 43.1 describe intrinsic properties of their subjects (e.g. *a jewel sparkles*, *a flag flutters*). Their similar alternations and PP SCFs make it difficult to separate them on syntactic grounds.

- **2** in class 51.3.2 (RUN verbs): 47.3 and 51.3.2 share the meaning component of motion. Their members take similar alternations and SCFs, which causes the confusion.

[2]Recall is not shown as it is identical here with accuracy.

[3]Recall that the amount of training data is smaller with resampling evaluation, see section 5.2.

	Feature set 1			Feature set 2			Feature set 3		
	ACC	P	F	ACC	P	F	ACC	P	F
RAND	5.8			5.8			5.8		
KNN	44.1	48.4	44.0	54.9	56.9	53.9	49.5	47.0	48.2
ME	47.5	49.4	47.6	59.3	61.4	59.9	59.3	61.9	60.0
SVM	47.1	50.4	47.8	57.8	59.4	57.9	57.8	60.1	58.2

Table 2: 'Leave one out' cross-validation results for KNN, ME, and SVM

	Feature set 1			Feature set 2			Feature set 3		
	ACC	P	F	ACC	P	F	ACC	P	F
RAND	5.8			5.8			5.8		
KNN	37.3	39.9	36.5	42.7	47.2	42.6	27.1	34.2	28.2
ME	47.1	47.3	47.0	58.1	59.1	58.1	60.1	60.5	59.8
SVM	47.3	50.2	47.7	56.8	59.5	57.1	54.4	56.5	54.6

Table 3: Re-sampling results for KNN, ME, and SVM

- **2** in class 37.7 (SAY verbs) and 1 in class 37.3 (MANNER OF SPEAKING verbs): 47.3 differs in semantics and syntax from 37.7 and 37.3. The confusion is due to idiosyncratic properties of individual verbs (e.g. *quake, wiggle*).

- **1** in class 36.1 (CORRESPOND verbs): 47.3 and 36.1 are semantically very different, but their members take similar intransitive and PP SCFs with high frequency.

- **1** in class 45.4 (OTHER CHANGE OF STATE verbs): Classes 47.3 and 45.3 are semantically different. Their similar PP SCFs explains the misclassification.

Most errors concern classes which are in fact semantically related. Unfortunately there is no gold standard which would comprehensively capture the semantic relatedness of Levin classes. Other errors concern semantically unrelated but syntactically similar classes – cases which we may be able to address in the future with careful feature engineering. Some errors relate to syntactic idiosyncracy. These show the true limits of lexical classification - the fact that the correspondence between the syntax and semantics of verbs is not always perfect.

6 Discussion and Conclusion

Our best results (e.g. 60.1 accuracy and 59.8 F-measure of ME) are good, considering that no sophisticated feature engineering / selection based on the properties of the target classification was performed in these experiments. The closest comparison point is the recent experiment reported by Joanis et al. (2007) which involved classifying 835 English verbs to 14 Levin classes using SVM. Features were specifically selected via analysis of alternations that are used to characterize Levin classes. Both shallow syntactic features (syntactic slots obtained using a chunker) and deep ones (SCFs extracted using Briscoe and Carroll's system) were used. The accuracy was 58% with the former and only 38% with the latter. This experiment is not directly comparable with ours as we classified a smaller number of verbs (204) to a higher number of Levin classes (17) (i.e. we had less training data) and did not select the optimal set of features using Levin's alternations. We nevertheless obtained better accuracy with our best performing method, and better accuracy (47%) with the same method (SVM) when the comparable feature set 1 was acquired using the very same subcategorization acquisition system.

It is likely that using larger and noisier SCF data explains the better result, suggesting that rich syntactic features incorporating information about both arguments and adjuncts are ideal for verb classification. Further experiments are required to determine the optimal set of features. In the future, we plan to experiment with different (noisy and filtered) versions of VALEX and add to the comparison a shallower set of features (e.g. NP and PP slots in VALEX regardless of the specific SCFs). We will also improve the features e.g. by enriching them with additional syntactic information available in VALEX lexical entries.

Acknowledgement

This work was partially supported by the Royal Society, UK.

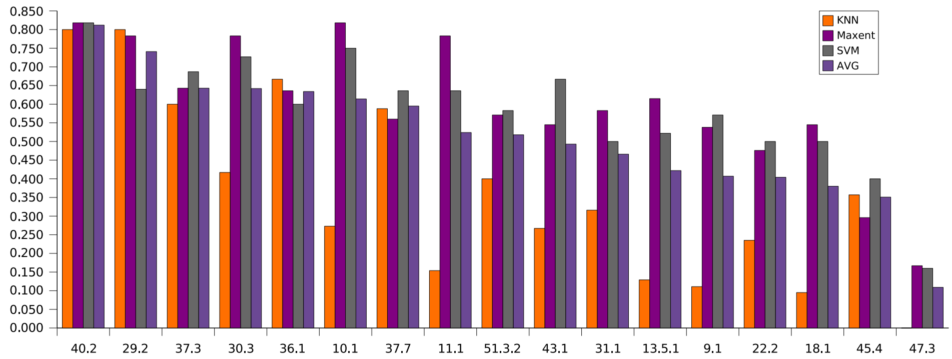


Figure 1: Class level F-score for feature set 3 (cross-validation)

References

E. J. Briscoe and J. Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the* 5^{th} *ACL Conference on Applied Natural Language Processing*, pages 356–363, Washington DC.

E. J. Briscoe and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the* 3^{rd} *LREC*, pages 1499–1504, Las Palmas, Gran Canaria.

C. Chang and J. Lin, 2001. *LIBSVM: a library for support vector machines*.

H. T. Dang. 2004. *Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*. Ph.D. thesis, CIS, University of Pennsylvania.

B. J. Dorr. 1997. Large-scale dictionary construction for foreign language tutoring and interlingual machine translation. *Machine Translation*, 12(4):271–322.

W. Hsu, C. Chang, and J. Lin. 2003. A practical guide to support vector classification.

R. Jackendoff. 1990. *Semantic Structures*. MIT Press, Cambridge, Massachusetts.

E. Joanis, S. Stevenson, and D. James. 2007. A general feature space for automatic verb classification. *Natural Language Engineering*, Forthcoming.

A. Korhonen, Y. Krymolowski, and Z. Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 64–71.

A. Korhonen, Y. Krymolowski, and T. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of LREC*.

B. Levin. 1993. *English Verb Classes and Alternations*. Chicago University Press, Chicago.

P. Merlo and S. Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27(3):373–408.

G. A. Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

S. D. Pietra, J. D. Pietra, and J. D. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

D. Prescher, S. Riezler, and M. Rooth. 2000. Using a probabilistic class-based lexicon for lexical ambiguity resolution. In *18th International Conference on Computational Linguistics*, pages 649–655, Saarbrücken, Germany.

S. Schulte im Walde. 2000. Clustering verbs semantically according to their alternation behaviour. In *Proceedings of COLING*, pages 747–753, Saarbrücken, Germany.

S. Schulte im Walde. 2006. Experiments on the automatic induction of german semantic verb classes. *Computational Linguistics*, 32(2):159–194.

L. Shi and R. Mihalcea. 2005. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, Mexico.

R. Swier and S. Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 95–102, Barcelona, Spain, August.

V. N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

L. Zhang, 2004. *Maximum Entropy Modeling Toolkit for Python and C++*, December.

MRD-based Word Sense Disambiguation: Further$_{\#2}$ Extending$_{\#1}$ Lesk

Timothy Baldwin,[♠] Su Nam Kim,[♠] Francis Bond,[♡] Sanae Fujita,[♢]
David Martinez[♠] and Takaaki Tanaka[♢]

♠ CSSE
University of Melbourne
VIC 3010 Australia

♡ NICT
3-5 Hikaridai, Seika-cho
Soraku-gun, Kyoto
619-0289 Japan

♢ NTT CS Labs
2-4 Hikari-dai, Seika-cho
Soraku-gun, Kyoto
619-0237 Japan

Abstract

This paper reconsiders the task of MRD-based word sense disambiguation, in extending the basic Lesk algorithm to investigate the impact on WSD performance of different tokenisation schemes, scoring mechanisms, methods of gloss extension and filtering methods. In experimentation over the Lexeed Sensebank and the Japanese Senseval-2 dictionary task, we demonstrate that character bigrams with sense-sensitive gloss extension over hyponyms and hypernyms enhances WSD performance.

1 Introduction

The aim of this work is to develop and extend word sense disambiguation (WSD) techniques to be applied to all words in a text. The goal of WSD is to link occurrences of ambiguous words in specific contexts to their meanings, usually represented by a machine readable dictionary (MRD) or a similar lexical repository. For instance, given the following Japanese input:

(1)	おとなしい	<u>犬</u>	を	飼いたい
	quiet	dog	ACC	want to keep

“(I) want to keep a quiet dog”

we would hope to identify each component word as occurring with the sense corresponding to the indicated English glosses.

WSD systems can be classified according to the knowledge sources they use to build their models. A top-level distinction is made between supervised and unsupervised systems. The former rely on training instances that have been hand-tagged, while the latter rely on other types of knowledge, such as lexical databases or untagged corpora. The Senseval evaluation tracks have shown that supervised systems perform better when sufficient training data is available, but they do not scale well to all words in context. This is known as the knowledge acquisition bottleneck, and is the main motivation behind research on unsupervised techniques (Mihalcea and Chklovski, 2003).

In this paper, we aim to exploit an existing lexical resource to build an all-words Japanese word-sense disambiguator. The resource in question is the Lexeed Sensebank (Tanaka et al., 2006) and consists of the 28,000 most familiar words of Japanese, each of which has one or more basic senses. The senses take the form of a dictionary definition composed from the closed vocabulary of the 28,000 words contained in the dictionary, each of which is further manually sense annotated according to the Lexeed sense inventory. Lexeed also has a semi-automatically constructed ontology.

Through the Lexeed sensebank, we investigate a number of areas of general interest to the WSD community. First, we test extensions of the Lesk algorithm (Lesk, 1986) over Japanese, focusing specifically on the impact of the overlap metric and segment representation on WSD performance. Second, we propose further extensions of the Lesk algorithm that make use of disambiguated definitions. In this, we shed light on the relative benefits we can expect from hand-tagging dictionary definitions, i.e. in introducing “semi-supervision” to the disambiguation task. The proposed method is language independent, and is equally applicable to the Extended WordNet[1] for English, for example.

2 Related work

Our work focuses on unsupervised and semi-supervised methods that target all words and parts of speech (POS) in context. We use the term “unsupervised” to refer to systems that do not use hand-tagged example sets for each word, in line with the standard usage in the WSD literature (Agirre and Edmonds, 2006). We blur the supervised/unsupervised boundary somewhat in combining the basic unsupervised methods with hand-tagged definitions from Lexeed, in order to measure the improvement we can expect from sense-tagged data. We qualify our use of hand-tagged definition

[1] http://xwn.hlt.utdallas.edu

sentences by claiming that this kind of resource is less costly to produce than sense-annotated open text because: (1) the effects of discourse are limited, (2) syntax is relatively simple, (3) there is significant semantic priming relative to the word being defined, and (4) there is generally explicit meta-tagging of the domain in technical definitions. In our experiments, we will make clear when hand-tagged sense information is being used.

Unsupervised methods rely on different knowledge sources to build their models. Primarily the following types of lexical resources have been used for WSD: MRDs, lexical ontologies, and untagged corpora (monolingual corpora, second language corpora, and parallel corpora). Although early approaches focused on exploiting a single resource (Lesk, 1986), recent trends show the benefits of combining different knowledge sources, such as hierarchical relations from an ontology and untagged corpora (McCarthy et al., 2004). In this summary, we will focus on a few representative systems that make use of different resources, noting that this is an area of very active research which we cannot do true justice to within the confines of this paper.

The Lesk method (Lesk, 1986) is an MRD-based system that relies on counting the overlap between the words in the target context and the dictionary definitions of the senses. In spite of its simplicity, it has been shown to be a hard baseline for unsupervised methods in Senseval, and it is applicable to all-words with minimal effort. Banerjee and Pedersen (2002) extended the Lesk method for WordNet-based WSD tasks, to include hierarchical data from the WordNet ontology (Fellbaum, 1998). They observed that the hierarchical relations significantly enhance the basic model. Both these methods will be described extensively in Section 3.1, as our approach is based on them.

Other notable unsupervised and semi-supervised approaches are those of McCarthy et al. (2004), who combine ontological relations and untagged corpora to automatically rank word senses in relation to a corpus, and Leacock et al. (1998) who use untagged data to build sense-tagged data automatically based on monosemous words. Parallel corpora have also been used to avoid the need for hand-tagged data, e.g. by Chan and Ng (2005).

3 Background

As background to our work, we first describe the basic and extended Lesk algorithms that form the core of our approach. Then we present the Lexeed lexical resource we have used in our experiments, and finally we outline aspects of Japanese relevant for this work.

3.1 Basic and Extended Lesk

The original Lesk algorithm (Lesk, 1986) performs WSD by calculating the relative word overlap between the context of usage of a target word, and the dictionary definition of each of its senses in a given MRD. The sense with the highest overlap is then selected as the most plausible hypothesis.

An obvious shortcoming of the original Lesk algorithm is that it requires that the exact words used in the definitions be included in each usage of the target word. To redress this shortcoming, Banerjee and Pedersen (2002) extended the basic algorithm for WordNet-based WSD tasks to include hierarchical information, i.e. expanding the definitions to include definitions of hypernyms and hyponyms of the synset containing a given sense, and assigning the same weight to the words sourced from the different definitions.

Both of these methods can be formalised according to the following algorithm, which also forms the basis of our proposed method:

---

for each word w_i in context $\mathbf{w} = w_1 w_2 ... w_n$ **do**
  for each sense $s_{i,j}$ and definition $\mathbf{d}_{i,j}$ of w_i **do**
    $score(s_{i,j}) = overlap(\mathbf{w}, \mathbf{d}_{i,j})$
  end for
  $s_i^* = \arg\max_j \, score(s_{i,j})$
end for

3.2 The Lexeed Sensebank

All our experimentation is based on the Lexeed Sensebank (Tanaka et al., 2006). The Lexeed Sensebank consists of all Japanese words above a certain level of familiarity (as defined by Kasahara et al. (2004)), giving rise to 28,000 words in all, with a total of 46,000 senses which are similarly filtered for similarity. The sense granularity is relatively coarse for most words, with the possible exception of light verbs, making it well suited to open-domain applications. Definition sentences for these senses were rewritten to use only the closed vocabulary of the 28,000 familiar words (and some function words). Additionally, a single example sentence was manually constructed to exemplify each of the 46,000 senses, once again using the closed vocabulary of the Lexeed dictionary. Both the definition sentences and example sentences were then manually sense annotated by 5 native speakers of Japanese, from which a majority sense was extracted.

In addition, an ontology was induced from the Lexeed dictionary, by parsing the first definition sentence for each sense (Nichols et al., 2005). Hypernyms were determined by identifying the highest scoping real predicate (i.e. the genus). Other relation types such as synonymy and domain were also induced based on trigger patterns in the definition sentences, although these are too few to be useful in our research. Because each word is sense tagged, the relations link senses rather than just words.

3.3 Peculiarities of Japanese

The experiments in this paper focus exclusively on Japanese WSD. Below, we outline aspects of Japanese which are relevant to the task.

First, Japanese is a non-segmenting language, i.e. there is no explicit orthographic representation of word boundaries. The native rendering of (1), e.g., is おとなしい犬を飼いたい. Various packages exist to automatically segment Japanese strings into words, and the Lexeed data has been pre-segmented using ChaSen (Matsumoto et al., 2003).

Second, Japanese is made up of 3 basic alphabets: hiragana, katakana (both syllabic in nature) and kanji (logographic in nature). The relevance of these first two observations to WSD is that we can choose to represent the context of a target word by way of characters or words.

Third, Japanese has relatively free word order, or strictly speaking, word order within phrases is largely fixed but the ordering of phrases governed by a given predicate is relatively free.

4 Proposed Extensions

We propose extensions to the basic Lesk algorithm in the orthogonal areas of the scoring mechanism, tokenisation, extended glosses and filtering.

4.1 Scoring Mechanism

In our algorithm, *overlap* provides the means to score a given pairing of context $\mathbf{w}$ and definition $\mathbf{d}_{i,j}$. In the original Lesk algorithm, *overlap* was simply the sum of words in common between the two, which Banerjee and Pedersen (2002) modified by squaring the size of each overlapping sub-string. While squaring is well motivated in terms of preferring larger substring matches, it makes the algorithm computationally expensive. We thus adopt a cheaper scoring mechanism which normalises relative to the length of $\mathbf{w}$ and $\mathbf{d}_{i,j}$, but ignores the length of substring matches. Namely, we use the Dice coefficient.

4.2 Tokenisation

Tokenisation is particularly important in Japanese because it is a non-segmenting language with a logographic orthography (kanji). As such, we can chose to either word tokenise via a word splitter such as ChaSen, or character tokenise. Character and word tokenisation have been compared in the context of Japanese information retrieval (Fujii and Croft, 1993) and translation retrieval (Baldwin, 2001), and in both cases, characters have been found to be the superior representation overall.

Orthogonal to the question of whether to tokenise into words or characters, we adopt an n-gram segment representation, in the form of simple unigrams and simple bigrams. In the case of word tokenisation and simple bigrams, e.g., example (1) would be represented as { おとなしい犬 , 犬を , を飼いたい }.

4.3 Extended Glosses

The main direction in which Banerjee and Pedersen (2002) successfully extended the Lesk algorithm was in including hierarchically-adjacent glosses (i.e. hyponyms and hypernyms). We take this a step further, in using both the Lexeed ontology and the sense-disambiguated words in the definition sentences.

The basic form of extended glossing is the simple Lesk method, where we take the simple definitions for each sense $s_{i,j}$ (i.e. without any gloss extension).

Next, we replicate the Banerjee and Pedersen (2002) method in extending the glosses to include words from the definitions for the (immediate) hypernyms and/or hyponyms of each sense $s_{i,j}$.

An extension of the Banerjee and Pedersen (2002) method which makes use of the sense-annotated definitions is to include the words in the definition of each sense-annotated word d_k contained in definition $\mathbf{d}_{i,j} = d_1 d_2 ... d_m$ of word sense $s_{i,j}$. That is, rather than traversing the ontology relative to each word sense candidate $s_{i,j}$ for the target word w_i, we represent each word sense via the original definition plus all definitions of word senses contained in it (weighting each to give the words in the original definition greater import than those from definitions of those word senses). We can then optionally adopt a similar policy to Banerjee and Pedersen (2002) in expanding each sense-annotated word d_k in the original definition relative to the ontology, to include the immediate hypernyms and/or hyponyms.

We further expand the definitions (+extdef) by adding the full definition for each sense-tagged word in the original definition. This can be combined with the Banerjee and Pedersen (2002) method by

also expanding each sense-annotated word d_k in the original definition relative to the ontology, to include the immediate hypernyms (+hyper) and/or hyponyms (+hypo).

4.4 Filtering

Each word sense in the dictionary is marked with a word class, and the word splitter similarly POS tags every definition and input to the system. It is natural to expect that the POS tag of the target word should match the word class of the word sense, and this provides a coarse-grained filter for discriminating homographs with different word classes.

We also experiment with a stop word-based filter which ignores a closed set of 18 lexicographic markers commonly found in definitions (e.g. 略 [*ryaku*] "an abbreviation for ..."), in line with those used by Nichols et al. (2005) in inducing the ontology.

5 Evaluation

We evaluate our various extensions over two datasets: (1) the example sentences in the Lexeed sensebank, and (2) the Senseval-2 Japanese dictionary task (Shirai, 2002).

All results below are reported in terms of simple precision, following the conventions of Senseval evaluations. For all experiments, precision and recall are identical as our systems have full coverage.

For the two datasets, we use two baselines: a random baseline and the first-sense baseline. Note that the first-sense baseline has been shown to be hard to beat for unsupervised systems (McCarthy et al., 2004), and it is considered supervised when, as in this case, the first-sense is the most frequent sense from hand-tagged corpora.

5.1 Lexeed Example Sentences

The goal of these experiments is to tag all the words that occur in the example sentences in the Lexeed Sensebank. The first set of experiments over the Lexeed Sensebank explores three parameters: the use of characters vs. words, unigrams vs. bigrams, and original vs. extended definitions. The results of the experiments and the baselines are presented in Table 1.

First, characters are in all cases superior to words as our segment granularity. The introduction of bigrams has a uniformly negative impact for both characters and words, due to the effects of data sparseness. This is somewhat surprising for characters, given that the median word length is 2 characters, although the difference between character unigrams and bigrams is slight.

Extended definitions are also shown to be superior to simple definitions, although the relative increment in making use of large amounts of sense annotations is smaller than that of characters vs. words, suggesting that the considerable effort in sense annotating the definitions is not commensurate with the final gain for this simple method.

Note that at this stage, our best-performing method is roughly equivalent to the unsupervised (random) baseline, but well below the supervised (first sense) baseline.

Having found that extended definitions improve results to a small degree, we turn to our next experiment were we investigate whether the introduction of ontological relations to expand the original definitions further enhances our precision. Here, we persevere with the use of word and characters (all unigrams), and experiment with the addition of hypernyms and/or hyponyms, with and without the extended definitions. We also compare our method directly with that of Banerjee and Pedersen (2002) over the Lexeed data, and further test the impact of the sense annotations, in rerunning our experiments with the ontology in a sense-*insensitive* manner, i.e. by adding in the union of word-level hypernyms and/or hyponyms. The results are described in Table 2. The results in brackets are reproduced from earlier tables.

Adding in the ontology makes a significant difference to our results, in line with the findings of Banerjee and Pedersen (2002). Hyponyms are better discriminators than hypernyms (assuming a given word sense has a hyponym – the Lexeed ontology is relatively flat), partly because while a given word sense will have (at most) one hypernym, it often has multiple hyponyms (if any at all). Adding in hypernyms or hyponyms, in fact, has a greater impact on results than simple extended definitions (+extdef), especially for words. The best overall results are produced for the (weighted) combination of all ontological relations (i.e. extended definitions, hypernyms and hyponyms), achieving a precision level above both the unsupervised (random) and supervised (first-sense) baselines.

In the interests of getting additional insights into the import of sense annotations in our method, we ran both the original Banerjee and Pedersen (2002) method and a sense-insensitive variant of our proposed method over the same data, the results for which are also included in Table 2. Simple hyponyms (without extended definitions) and word-based segments returned the best results out of all the variants tried, at a precision of 0.656. This compares with a precision of 0.683 achieved for the best

	UNIGRAMS		BIGRAMS	
	ALL WORDS	POLYSEMOUS	ALL WORDS	POLYSEMOUS
Simple Definitions				
CHARACTERS	0.523	0.309	0.486	0.262
WORDS	0.469	0.229	0.444	0.201
Extended Definitions				
CHARACTERS	**0.526**	**0.313**	0.529	0.323
WORDS	0.489	0.258	0.463	0.227

Table 1: Precision over the Lexeed example sentences using simple/extended definitions and word/character unigrams and bigrams (best-performing method in **boldface**)

		ALL WORDS	POLYSEMOUS
UNSUPERVISED BASELINE:		0.527	0.315
SUPERVISED BASELINE:		0.633	0.460
Banerjee and Pedersen (2002)		0.648	0.492
Ontology expansion (sense-sensitive)			
W	simple	(0.469)	(0.229)
	+extdef	(0.489)	(0.258)
	+hypernyms	0.559	0.363
	+hyponyms	0.655	0.503
	+def +hyper	0.577	0.386
	+def +hypo	0.649	0.490
	+def +hyper +hypo	**0.683**	**0.539**
C	simple	(0.523)	(0.309)
	+extdef	(0.526)	(0.313)
	+hypernyms	0.539	0.334
	+hyponyms	0.641	0.481
	+def +hyper	0.563	0.365
	+def +hypo	0.671	0.522
	+def +hyper +hypo	0.671	0.522
Ontology expansion (sense-insensitive)			
W	+hypernyms	0.548	0.348
	+hyponyms	0.656	0.503
	+def +hyper	0.551	0.347
	+def +hypo	0.649	0.490
	+def + hyper +hypo	0.631	0.464
C	+hypernyms	0.537	0.332
	+hyponyms	0.644	0.485
	+def +hyper	0.542	0.335
	+def +hypo	0.644	0.484
	+def + hyper +hypo	0.628	0.460

Table 2: Precision over the Lexeed example sentences using ontology-based gloss extension (with/without word sense information) and word (W) and character (C) unigrams (best-performing method in **boldface**)

of the sense-sensitive methods, indicating that sense information enhances WSD performance. This reinforces our expectation that richly annotated lexical resources improve performance. With richer information to work with, character based methods uniformly give worse results.

While we don't present the results here due to reasons of space, POS-based filtering had very little impact on results, due to very few POS-differentiated homographs in Japanese. Stop word filtering leads to a very slight increment in precision across the board (of the order of 0.001).

		ALL WORDS	POLYSEMOUS
Baselines			
	Unsupervised (random)	0.310	0.260
	Supervised (first-sense)	0.577	0.555
Ontology expansion (sense-sensitive)			
W	+def +hyper +hypo	**0.624**	**0.605**
C	+def +hyper +hypo	**0.624**	**0.605**
Ontology expansion (sense-insensitive)			
W	+def +hyper +hypo	0.602	0.581
C	+def +hyper +hypo	0.593	0.572

Table 3: Precision over the Senseval-2 data

5.2 Senseval-2 Japanese Dictionary Task

In our second set of experiments we apply our proposed method to the Senseval-2 Japanese dictionary task (Shirai, 2002) in order to calibrate our results against previously published results for Japanese WSD. Recall that this is a lexical sample task, and that our evaluation is relative to Lexeed reannotations of the same dataset, although the relative polysemy for the original data and the re-annotated version are largely the same (Tanaka et al., 2006). The first sense baselines (i.e. sense skewing) for the two sets of annotations differ significantly, however, with a precision of 0.726 reported for the original task, and 0.577 for the re-annotated Lexeed variant. System comparison (Senseval-2 systems vs. our method) will thus be reported in terms of error rate reduction relative to the respective first sense baselines.

In Table 3, we present the results over the Senseval-2 data for the best-performing systems from our earlier experiments. As before, we include results over both words and characters, and with sense-sensitive and sense-insensitive ontology expansion.

Our results largely mirror those of Table 2, although here there is very little to separate words and characters. All methods surpassed both the random and first sense baselines, but the relative impact

of sense annotations was if anything even less pronounced than for the example sentence task.

Both sense-sensitive WSD methods achieve a precision of 0.624 over all the target words (with one target word per sentence), an error reduction rate of 11.1%. This compares favourably with an error rate reduction of 21.9% for the best of the WSD systems in the original Senseval-2 task (Kurohashi and Shirai, 2001), particularly given that our method is semi-supervised while the Senseval-2 system is a conventional supervised word sense disambiguator.

6 Conclusion

In our experiments extending the Lesk algorithm over Japanese data, we have shown that definition expansion via an ontology produces a significant performance gain, confirming results by Banerjee and Pedersen (2002) for English. We also explored a new expansion of the Lesk method, by measuring the contribution of sense-tagged definitions to overall disambiguation performance. Using sense information doubles the error reduction compared to the supervised baseline, a constant gain that shows the importance of precise sense information for error reduction.

Our WSD system can be applied to all words in running text, and is able to improve over the first-sense baseline for two separate WSD tasks, using only existing Japanese resources. This full-coverage system opens the way to explore further enhancements, such as the contribution of extra sense-tagged examples to the expansion, or the combination of different WSD algorithms.

For future work, we are also studying the integration of the WSD tool with other applications that deal with Japanese text, such as a cross-lingual glossing tool that aids Japanese learners reading text. Another application we are working on is the integration of the WSD system with parse selection for Japanese grammars.

Acknowledgements

This material is supported by the Research Collaboration between NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation and the University of Melbourne. We would like to thank members of the NTT Machine Translation Group and the three anonymous reviewers for their valuable input on this research.

References

Eneko Agirre and Philip Edmonds, editors. 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer, Dordrecht, Netherlands.

Timothy Baldwin. 2001. Low-cost, high-performance translation retrieval: Dumber is better. In *Proc. of the 39th Annual Meeting of the ACL and 10th Conference of the EACL (ACL-EACL 2001)*, pages 18–25, Toulouse, France.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 136–45, Mexico City, Mexico.

Yee Seng Chan and Hwee Tou Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1037–42, Pittsburgh, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Hideo Fujii and W. Bruce Croft. 1993. A comparison of indexing techniques for Japanese text retrieval. In *Proc. of 16th International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 237–46, Pittsburgh, USA.

Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. 2004. Construction of a Japanese semantic lexicon: Lexeed. In *Proc. of SIG NLC-159*, Tokyo, Japan.

Sadao Kurohashi and Kiyoaki Shirai. 2001. SENSEVAL-2 Japanese tasks. In *IEICE Technical Report NLC 2001-10*, pages 1–8. (in Japanese).

Claudia Leacock, Martin Chodorow, and George A. Miller. 1998. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24(1):147–65.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proc. of the 1986 SIGDOC Conference*, pages 24–6, Ontario, Canada.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Hiroshi Matsuda, Kazuma Takaoka, and Masayuki Asahara. 2003. *Japanese Morphological Analysis System ChaSen Version 2.3.3 Manual*. Technical report, NAIST.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proc. of the 42nd Annual Meeting of the ACL*, pages 280–7, Barcelona, Spain.

Rada Mihalcea and Timothy Chklovski. 2003. Open Mind Word Expert: Creating Large Annotated Data Collections with Web Users' Help. In *Proceedings of the EACL 2003 Workshop on Linguistically Annotated Corpora (LINC 2003)*, pages 53–61, Budapest, Hungary.

Eric Nichols, Francis Bond, and Daniel Flickinger. 2005. Robust ontology acquisition from machine-readable dictionaries. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pages 1111–6, Edinburgh, UK.

Kiyoaki Shirai. 2002. Construction of a word sense tagged corpus for SENSEVAL-2 japanese dictionary task. In *Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, pages 605–8, Las Palmas, Spain.

Takaaki Tanaka, Francis Bond, and Sanae Fujita. 2006. The Hinoki sensebank — a large-scale word sense tagged corpus of Japanese —. In *Proc. of the Workshop on Frontiers in Linguistically Annotated Corpora 2006*, pages 62–9, Sydney, Australia.

Fast Computing Grammar-driven Convolution Tree Kernel for Semantic Role Labeling

Wanxiang Che[1]*, Min Zhang[2], Ai Ti Aw[2], Chew Lim Tan[3], Ting Liu[1], Sheng Li[1]
[1]School of Computer Science and Technology
Harbin Institute of Technology, China 150001
{car,tliu}@ir.hit.edu.cn, lisheng@hit.edu.cn
[2]Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
{mzhang,aaiti}@i2r.a-star.edu.sg
[3]School of Computing
National University of Singapore, Singapore 117543
tancl@comp.nus.edu.sg

Abstract

Grammar-driven convolution tree kernel (GTK) has shown promising results for semantic role labeling (SRL). However, the time complexity of computing the GTK is exponential in theory. In order to speed up the computing process, we design two fast grammar-driven convolution tree kernel (FGTK) algorithms, which can compute the GTK in polynomial time. Experimental results on the CoNLL-2005 SRL data show that our two FGTK algorithms are much faster than the GTK.

1 Introduction

Given a sentence, the task of semantic role labeling (SRL) is to analyze the propositions expressed by some target verbs or nouns and some constituents of the sentence. In previous work, data-driven techniques, including feature-based and kernel-based learning methods, have been extensively studied for SRL (Carreras and Màrquez, 2005).

Although feature-based methods are regarded as the state-of-the-art methods and achieve much success in SRL, kernel-based methods are more effective in capturing structured features than feature-based methods. In the meanwhile, the syntactic structure features hidden in a parse tree have been suggested as an important feature for SRL and need to be further explored in SRL (Gildea and Palmer, 2002; Punyakanok et al., 2005). Moschitti (2004) and Che et al. (2006) are two reported work to use convolution tree kernel (TK) methods (Collins and Duffy, 2001) for SRL and has shown promising results. However, as a general learning algorithm, the TK only carries out hard matching between two subtrees without considering any linguistic knowledge in kernel design. To solve the above issue, Zhang et al. (2007) proposed a grammar-driven convolution tree kernel (GTK) for SRL. The GTK can utilize more grammatical structure features via two grammar-driven approximate matching mechanisms over substructures and nodes. Experimental results show that the GTK significantly outperforms the TK (Zhang et al., 2007). Theoretically, the GTK method is applicable to any problem that uses syntax structure features and can be solved by the TK methods, such as parsing, relation extraction, and so on. In this paper, we use SRL as an application to test our proposed algorithms.

Although the GTK shows promising results for SRL, one big issue for the kernel is that it needs exponential time to compute the kernel function since it need to explicitly list all the possible variations of two sub-trees in kernel calculation (Zhang et al., 2007). Therefore, this method only works efficiently on such kinds of datasets where there are not too many optional nodes in production rule set. In order to solve this computation issue, we propose two fast algorithms to compute the GTK in polynomial time.

The remainder of the paper is organized as follows: Section 2 introduces the GTK. In Section 3, we present our two fast algorithms for computing the GTK. The experimental results are shown in Section 4. Finally, we conclude our work in Section 5.

*The work was mainly done when the author was a visiting student at I^{2}R

2 Grammar-driven Convolution Tree Kernel

The GTK features with two grammar-driven approximate matching mechanisms over substructures and nodes.

2.1 Grammar-driven Approximate Matching

Grammar-driven Approximate Substructure Matching: the TK requires exact matching between two phrase structures. For example, the two phrase structures "NP→DT JJ NN" (NP→*a red car*) and "NP→DT NN" (NP→*a car*) are not identical, thus they contribute nothing to the conventional kernel although they share core syntactic structure property and therefore should play the same semantic role given a predicate. Zhang et al. (2007) introduces the concept of optional node to capture this phenomenon. For example, in the production rule "NP→DT [JJ] NP", where [JJ] denotes an optional node. Based on the concept of optional node, the grammar-driven approximate substructure matching mechanism is formulated as follows:

$$M(r_1, r_2) = \sum_{i,j}(I_T(T^i_{r_1}, T^j_{r_2}) \times \lambda_1^{a_i+b_j}) \qquad (1)$$

where r_1 is a production rule, representing a two-layer sub-tree, and likewise for r_2. $T^i_{r_1}$ is the i^{th} variation of the sub-tree r_1 by removing one ore more optional nodes, and likewise for $T^j_{r_2}$. $I_T(\cdot,\cdot)$ is a binary function that is 1 iff the two sub-trees are identical and zero otherwise. λ_1 ($0 \leq \lambda_1 \leq 1$) is a small penalty to penalize optional nodes. a_i and b_j stand for the numbers of occurrence of removed optional nodes in subtrees $T^i_{r_1}$ and $T^j_{r_2}$, respectively.

$M(r_1, r_2)$ returns the similarity (i.e., the kernel value) between the two sub-trees r_1 and r_2 by summing up the similarities between all possible variations of the sub-trees.

Grammar-driven Approximate Node Matching: the TK needs an exact matching between two nodes. But, some similar POSs may represent similar roles, such as NN (*dog*) and NNS (*dogs*). Zhang et al. (2007) define some equivalent nodes that can match each other with a small penalty λ_2 ($0 \leq \lambda_2 \leq 1$). This case is called node feature *mutation*. The approximate node matching can be formulated as:

$$M(f_1, f_2) = \sum_{i,j}(I_f(f^i_1, f^j_2) \times \lambda_2^{a_i+b_j}) \qquad (2)$$

where f_1 is a node feature, f^i_1 is the i^{th} mutation of f_1 and a_i is 0 iff f^i_1 and f_1 are identical and 1 otherwise, and likewise for f_2 and b_j. $I_f(\cdot,\cdot)$ is a function that is 1 iff the two features are identical and zero otherwise. Eq. (2) sums over all combinations of feature mutations as the node feature similarity.

2.2 The GTK

Given these two approximate matching mechanisms, the GTK is defined by beginning with the feature vector representation of a parse tree T as:

$$\Phi'(T) = (\#subtree_1(T), \ldots, \#subtree_n(T))$$

where $\#subtree_i(T)$ is the occurrence number of the i^{th} sub-tree type ($subtree_i$) in T. Now the GTK is defined as follows:

$$\begin{aligned} K_G(T_1, T_2) &= \langle \Phi'(T_1), \Phi'(T_2) \rangle \\ &= \textstyle\sum_i \#subtree_i(T_1) \cdot \#subtree_i(T_2) \\ &= \textstyle\sum_i ((\sum_{n_1 \in N_1} I'_{subtree_i}(n_1)) \\ &\quad \cdot (\textstyle\sum_{n_2 \in N_2} I'_{subtree_i}(n_2))) \\ &= \textstyle\sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \Delta'(n_1, n_2) \end{aligned} \qquad (3)$$

where N_1 and N_2 are the sets of nodes in trees T_1 and T_2, respectively. $I'_{subtree_i}(n)$ is a function that is $\lambda_1^a \cdot \lambda_2^b$ iff there is a $subtree_i$ rooted at node n and zero otherwise, where a and b are the numbers of removed optional nodes and mutated node features, respectively. $\Delta'(n_1, n_2)$ is the number of the common $subtrees$ rooted at n_1 and n_2, i.e.,

$$\Delta'(n_1, n_2) = \sum_i I'_{subtree_i}(n_1) \cdot I'_{subtree_i}(n_2) \quad (4)$$

$\Delta'(n_1, n_2)$ can be further computed by the following recursive rules:

R-A: if n_1 and n_2 are pre-terminals, then:

$$\Delta'(n_1, n_2) = \lambda \times M(f_1, f_2) \qquad (5)$$

where f_1 and f_2 are features of nodes n_1 and n_2 respectively, and $M(f_1, f_2)$ is defined in Eq. (2), which can be computed in linear time $O(n)$, where n is the number of feature mutations.

R-B: else if both n_1 and n_2 are the same non-terminals, then generate all variations of sub-trees of *depth one* rooted at n_1 and n_2 (denoted by T_{n_1}

and T_{n_2} respectively) by removing different optional nodes, then:

$$\Delta'(n_1, n_2) = \lambda \times \sum_{i,j} I_T(T^i_{n_1}, T^j_{n_2}) \times \lambda_1^{a_i + b_j} \times \prod_{k=1}^{nc(n_1,i)} (1 + \Delta'(ch(n_1, i, k), ch(n_2, j, k))) \quad (6)$$

where $T^i_{n_1}$, $T^j_{n_2}$, $I_T(\cdot,\cdot)$, a_i and b_j have been explained in Eq. (1). $nc(n_1, i)$ returns the number of children of n_1 in its i^{th} subtree variation $T^i_{n_1}$. $ch(n_1, i, k)$ is the k^{th} child of node n_1 in its i^{th} variation subtree $T^i_{n_1}$, and likewise for $ch(n_2, j, k)$. λ ($0 < \lambda < 1$) is the decay factor.

R-C: else $\Delta'(n_1, n_2) = 0$

3 Fast Computation of the GTK

Clearly, directly computing Eq. (6) requires exponential time, since it needs to sum up all possible variations of the sub-trees with and without optional nodes. For example, supposing n_1 = "A→a [b] c [d]", n_2 = "A→a b c". To compute the Eq. (6), we have to list all possible variations of n_1 and n_2's subtrees, n_1: "A→a b c d", "A→a b c", "A→a c d", "A→a c"; n_2: "A→a b c". Unfortunately, Zhang et al. (2007) did not give any theoretical solution for the issue of exponential computing time. In this paper, we propose two algorithms to calculate it in polynomial time. Firstly, we recast the issue of computing Eq. (6) as a problem of finding common sub-trees with and without optional nodes between two subtrees. Following this idea, we rewrite Eq. (6) as:

$$\Delta'(n_1, n_2) = \lambda \times (1 + \sum_{p=lx}^{lm} \Delta_p(c_{n_1}, c_{n_2})) \quad (7)$$

where c_{n_1} and c_{n_2} are the child node sequences of n_1 and n_2, Δ_p evaluates the number of common sub-trees with exactly p children (at least including all non-optional nodes) rooted at n_1 and n_2, $lx = \max\{np(c_{n_1}), np(c_{n_2})\}$ and $np(\cdot)$ is the number of non-optional nodes, $lm = \min\{l(c_{n_1}), l(c_{n_2})\}$and $l(\cdot)$ returns the number of children.

Now let's study how to calculate $\Delta_p(c_{n_1}, c_{n_2})$ using dynamic programming algorithms. Here, we present two dynamic programming algorithms to compute it in polynomial time.

3.1 Fast Grammar-driven Convolution Tree Kernel I (FGTK-I)

Our FGTK-I algorithm is motivated by the string subsequence kernel (SSK) (Lodhi et al., 2002). Given two child node sequences $sx = c_{n_1}$ and $t = c_{n_2}$ (x is the last child), the SSK uses the following recursive formulas to evaluate the Δ_p:

$$\Delta'_0(s,t) = 1, \text{for all } s, t,$$

$$\Delta'_p(s,t) = 0, \text{if} \min(|s|,|t|) < p, \quad (8)$$

$$\Delta_p(s,t) = 0, \text{if} \min(|s|,|t|) < p, \quad (9)$$

$$\Delta'_p(sx,t) = \mu \times \Delta'_p(sx,t) + \sum_{j:t_j=x} (\Delta'_{p-1}(s, t[1:j-1] \times \mu^{|t|-j+2})), \quad (10)$$

$$p = 1, \ldots, n-1,$$

$$\Delta_p(sx,t) = \Delta_p(s,t) + \sum_{j:t_j=x} (\Delta'_{p-1}(s, t[1:j-1] \times \mu^2)). \quad (11)$$

where Δ'_p is an auxiliary function since it is only the interior gaps in the subsequences that are penalized; μ is a decay factor only used in the SSK for weighting each extra length unit. Lodhi et al. (2002) explained the correctness of the recursion defined above.

Compared with the SSK kernel, the GTK has three different features:

f1: In the GTK, only optional nodes can be skipped while the SSK kernel allows any node skipping;

f2: The GTK penalizes ***skipped optional*** nodes only (including both interior and exterior skipped nodes) while the SSK kernel weights the length of subsequences (all interior skipped nodes are counted in, but exterior nodes are ignored);

f3: The GTK needs to further calculate the number of common sub-trees rooted at each two matching node pair x and $t[j]$.

To reflect the three considerations, we modify the SSK kernel as follows to calculate the GTK:

$$\Delta_0(s,t) = opt(s) \times opt(t) \times \lambda_1^{|s|+|t|}, \text{for all } s, t, \quad (12)$$

$$\Delta_p(s,t) = 0, \text{if} \min(|s|,|t|) < p, \quad (13)$$

$$\Delta_p(sx,t) = \lambda_1 \times \Delta_p(sx,t) \times opt(x) + \sum_{j:t_j=x} (\Delta_{p-1}(s, t[1:j-1]) \times \lambda^{|t|-j} \times opt(t[j+1:|t|]) \times \Delta'(x, t[j])). \quad (14)$$

where $opt(w)$ is a binary function, which is 0 if non-optional nodes are found in the node sequence w and 1 otherwise (*f1*); λ_1 is the penalty to penalize skipped optional nodes and the power of λ_1 is the number of skipped optional nodes (*f2*); $\Delta'(x, t[j])$ is defined in Eq. (7) (*f3*). Now let us compare

the FGTK-I and SSK kernel algorithms. Based on Eqs. (8), (9), (10) and (11), we introduce the $opt(\cdot)$ function and the penalty λ_1 into Eqs. (12), (13) and (14), respectively. $opt(\cdot)$ is to ensure that in the GTK only optional nodes are allowed to be skipped. And only those skipped optional nodes are penalized with λ_1. Please note that Eqs. (10) and (11) are merged into Eq. (14) because of the different meaning of μ and λ_1. From Eq. (8), we can see that the current path in the recursive call will stop and its value becomes zero once non-optional node is skipped (when $opt(w) = 0$).

Let us use a sample of n_1 = "A→a [b] c [d]", n_2 = "A→a b c" to exemplify how the FGTK-I algorithm works. In Eq. (14)'s vocabulary, we have s = "a [b] c", t = "a b c", x = "[d]", $opt(x) = opt([d]) = 1$, $p = 3$. Then according to Eq (14), $\Delta_p(c_{n_1}, c_{n_2})$ can be calculated recursively as Eq. (15) (Please refer to the next page).

Finally, we have $\Delta_p(c_{n_1}, c_{n_2}) = \lambda_1 \times \Delta'(\text{a, a}) \times \Delta'(\text{b, b}) \times \Delta'(\text{c, c})$

By means of the above algorithm, we can compute the $\Delta'(n_1, n_2)$ in $O(p|c_{n_1}| \cdot |c_{n_2}|^2)$ (Lodhi et al., 2002). This means that the worst case complexity of the FGTK-I is $O(p\rho^3|N_1| \cdot |N_2|^2)$, where ρ is the maximum branching factor of the two trees.

3.2 Fast Grammar-driven Convolution Tree Kernel II (FGTK-II)

Our FGTK-II algorithm is motivated by the partial trees (PTs) kernel (Moschitti, 2006). The PT kernel algorithm uses the following recursive formulas to evaluate $\Delta_p(c_{n_1}, c_{n_2})$:

$$\Delta_p(c_{n_1}, c_{n_2}) = \sum_{i=1}^{|c_{n_1}|} \sum_{j=1}^{|c_{n_2}|} \Delta'_p(c_{n_1}[1:i], c_{n_2}[1:j]) \quad (16)$$

where $c_{n_1}[1:i]$ and $c_{n_2}[1:j]$ are the child subsequences of c_{n_1} and c_{n_2} from 1 to i and from 1 to j, respectively. Given two child node sequences $s_1a = c_{n_1}[1:i]$ and $s_2b = c_{n_2}[1:j]$ (a and b are the last children), the PT kernel computes $\Delta'_p(\cdot, \cdot)$ as follows:

$$\Delta'_p(s_1a, s_2b) = \begin{cases} \mu^2\Delta'(a,b)D_p(|s_1|, |s_2|) & \text{if } a = b \\ 0 & \text{else} \end{cases} \quad (17)$$

where $\Delta'(a, b)$ is defined in Eq. (7) and D_p is recursively defined as follows:

$$\begin{aligned} D_p(k,l) = {} & \Delta'_{p-1}(s_1[1:k], s_2[1:l]) \\ & + \mu D_p(k, l-1) + \mu D_p(k-1, l) \quad (18) \\ & - \mu^2 D_p(k-1, l-1) \end{aligned}$$

$$D_1(k,l) = 1, \text{for all } k, l \quad (19)$$

where μ used in Eqs. (17) and (18) is a factor to penalize the length of the child sequences.

Compared with the PT kernel, the GTK has two different features which are the same as *f1* and *f2* when defining the FGTK-I.

To reflect the two considerations, based on the PT kernel algorithm, we define another fast algorithm of computing the GTK as follows:

$$\begin{aligned} \Delta_p(c_{n_1}, c_{n_2}) = {} & \textstyle\sum_{i=1}^{|c_{n_1}|} \sum_{j=1}^{|c_{n_2}|} \Delta'_p(c_{n_1}[1:i], c_{n_2}[1:j]) \\ & \times opt(c_{n_1}[i+1 : |c_{n_1}|]) \\ & \times opt(c_{n_2}[j+1 : |c_{n_2}|]) \\ & \times \lambda_1^{|c_{n_1}|-i+|c_{n_2}|-j} \end{aligned} \quad (20)$$

$$\Delta'_p(s_1a, s_2b) = \begin{cases} \Delta'(a,b)D_p(|s_1|, |s_2|) & \text{if } a = b \\ 0 & \text{else} \end{cases} \quad (21)$$

$$\begin{aligned} D_p(k,l) = {} & \Delta'_{p-1}(s_1[1:k], s_2[1:l]) \\ & + \lambda_1 D_p(k, l-1) \times opt(s_2[l]) \quad (22) \\ & + \lambda_1 D_p(k-1, l) \times opt(s_1[k]) \\ & - \lambda_1^2 D_p(k-1, l-1) \times opt(s_1[k]) \times opt(s_2[l]) \end{aligned}$$

$$\begin{aligned} D_1(k,l) = {} & \lambda_1^{k+l} \times opt(s_1[1:k]) \times opt(s_2[1:l]), \quad (23) \\ & \text{for all } k, l \end{aligned}$$

$$\Delta'_p(s_1, s_2) = 0, \text{if } \min(|s_1|, |s_2|) < p \quad (24)$$

where $opt(w)$ and λ_1 are the same as them in the FGTK-I.

Now let us compare the FGTK-II and the PT algorithms. Based on Eqs. (16), (18) and (19), we introduce the $opt(\cdot)$ function and the penalty λ_1 into Eqs. (20), (22) and (23), respectively. This is to ensure that in the GTK only optional nodes are allowed to be skipped and only those skipped optional nodes are penalized. In addition, compared with Eq. (17), the penalty μ^2 is removed in Eq. (21) in view that our kernel only penalizes skipped nodes. Moreover, Eq. (24) is only for fast computing. Finally, the same as the FGTK-I, in the FGTK-II the current path in a recursive call will stop and its value becomes zero once non-optional node is skipped (when $opt(w) = 0$). Here, we still can use an example to derivate the process of the algorithm step by step as that for FGTK-I algorithm. Due to space limitation, here, we do not illustrate it in detail.

By means of the above algorithms, we can compute the $\Delta'(n_1, n_2)$ in $O(p|c_{n_1}| \cdot |c_{n_2}|)$ (Moschitti,

$$
\begin{aligned}
\Delta_p(c_{n_1}, c_{n_2}) &= \Delta_p(\text{“a [b] c [d]”}, \text{“a b c”}) \\
&= \lambda_1 \times \Delta_p(\text{“a [b] c”}, \text{“a b c”}) + 0 && \text{//Since } x \not\subseteq t\text{, the second term is 0} \\
&= \lambda_1 \times (0 + \Delta_{p-1}(\text{“a [b]”}, \text{“a b”}) \times \lambda_1^{3-3} \times \Delta'(c,c)) && \text{//Since } opt(\text{“c”}) = 0\text{, the first term is 0} \\
&= \lambda_1 \times \Delta'(c,c) \times (0 + \Delta_{p-2}(\text{“a”}, \text{“a b”}) \times \lambda_1^{2-2} \times \Delta'(\text{b},\text{b})) && \text{//Since } p-1 > |\text{“a”}|, \Delta_{p-2}(\text{“a”}, \text{“a b”}) = 0 \\
&= \lambda_1 \times \Delta'(c,c) \times (0 + \Delta'(\text{a},\text{a}) \times \Delta'(\text{b},\text{b})) && \text{//}\Delta_{p-2}(\text{“a”}, \text{“a”}) = \Delta'(\text{a},\text{a})
\end{aligned}
\tag{15}
$$

2006). This means that the worst complexity of the FGTK-II is $O(p\rho^2|N_1| \cdot |N_2|)$. It is faster than the FGTK-I's $O(p\rho^3|N_1| \cdot |N_2|^2)$ in theory. Please note that the average ρ in natural language parse trees is very small and the overall complexity of the FGTKs can be further reduced by avoiding the computation of node pairs with different labels (Moschitti, 2006).

4 Experiments

4.1 Experimental Setting

Data: We use the CoNLL-2005 SRL shared task data (Carreras and Màrquez, 2005) as our experimental corpus.

Classifier: SVM (Vapnik, 1998) is selected as our classifier. In the FGTKs implementation, we modified the binary Tree Kernels in SVM-Light Tool (SVM-Light-TK) (Moschitti, 2006) to a grammar-driven one that encodes the GTK and the two fast dynamic algorithms inside the well-known SVM-Light tool (Joachims, 2002). The parameters are the same as Zhang et al. (2007).

Kernel Setup: We use Che et al. (2006)'s hybrid convolution tree kernel (the best-reported method for kernel-based SRL) as our baseline kernel. It is defined as $K_{hybrid} = \theta K_{path} + (1-\theta)K_{cs}$ $(0 \leq \theta \leq 1)$[1]. Here, we use the GTK to compute the K_{path} and the K_{cs}.

In the training data (WSJ sections 02-21), we get 4,734 production rules which appear at least 5 times. Finally, we use 1,404 rules with optional nodes for the approximate structure matching. For the node approximate matching, we use the same equivalent node sets as Zhang et al. (2007).

4.2 Experimental Results

We use 30,000 instances (a subset of the entire training set) as our training set to compare the different kernel computing algorithms [2]. All experiments are conducted on a PC with CPU 2.8GH and memory 1G. Fig. 1 reports the experimental results, where training curves (time vs. # of instances) of five kernels are illustrated, namely the TK, the FGTK-I, the FGTK-II, the GTK and a polynomial kernel (only for reference). It clearly demonstrates that our FGTKs are faster than the GTK algorithm as expected. However, the improvement seems not so significant. This is not surprising as there are only 30.4% rules (1,404 out of 4,734)[3] that have optional nodes and most of them have only one optional node[4]. Therefore, in this case, it is not time consuming to list all the possible sub-tree variations and sum them up. Let us study this issue from computational complexity viewpoint. Suppose all rules have exactly one optional node. This means each rule can only generate two variations. Therefore computing Eq. (6) is only 4 times (2*2) slower than the GTK in this case. In other words, we can say that given the constraint that there is only one optional node in one rule, the time complexity of the GTK is also $O(|N_1| \cdot |N_2|)$ [5], where N_1 and N_2 are the numbers of tree nodes, the same as the TK.

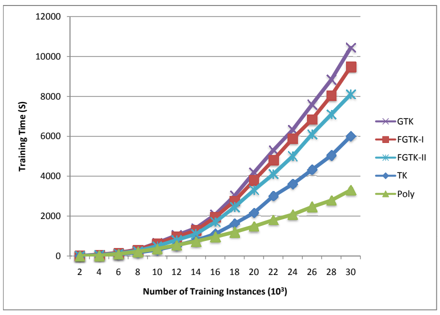


Figure 1: Training time comparison among different kernels with rule set having less optional nodes.

Moreover, Fig 1 shows that the FGTK-II is faster than the FGTK-I. This is reasonable since as dis-

[1] K_{path} and K_{cs} are two TKs to describe predicate-argument link features and argument syntactic structure features, respectively. For details, please refer to (Che et al., 2006).

[2] There are about 450,000 identification instances are extracted from training data.

[3] The percentage is even smaller if we consider all production (it becomes 14.4% (1,404 out of 9,700)).

[4] There are 1.6 optional nodes in each rule averagely.

[5] Indeed it is $O(4 \cdot |N_1| \cdot |N_2|)$. The parameter 4 is omitted when discussing time complexity.

cussed in Subsection 3.2, the FGTK-I's time complexity is $O(p\rho^3|N_1| \cdot |N_2|^2)$ while the FGTK-II's is $O(p\rho^2|N_1| \cdot |N_2|)$.

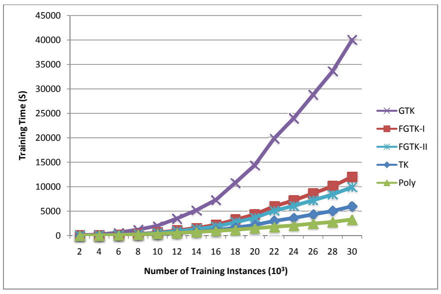


Figure 2: Training time comparison among different kernels with rule set having more optional nodes.

To further verify the efficiency of our proposed algorithm, we conduct another experiment. Here we use the same setting as that in Fig 1 except that we randomly add more optional nodes in more production rules. Table 1 reports the statistics on the two rule set. Similar to Fig 1, Fig 2 compares the training time of different algorithms. We can see that Fig 2 convincingly justify that our algorithms are much faster than the GTK when the experimental data has more optional nodes and rules.

Table 1: The rule set comparison between two experiments.

	# rules	# rule with at least optional nodes	# optional nodes	# average optional nodes per rule
Exp1	4,734	1,404	2,242	1.6
Exp2	4,734	4,520	10,451	2.3

5 Conclusion

The GTK is a generalization of the TK, which can capture more linguistic grammar knowledge into the later and thereby achieve better performance. However, a biggest issue for the GTK is its computing speed, which needs exponential time in theory. Therefore, in this paper we design two fast grammar-driven convolution tree kennel (FGTK-I and II) algorithms which can compute the GTK in polynomial time. The experimental results show that the FGTKs are much faster than the GTK when data set has more optional nodes. We conclude that our fast algorithms enable the GTK kernel to easily scale to larger dataset. Besides the GTK, the idea of our fast algorithms can be easily used into other similar problems.

To further our study, we will use the FGTK algorithms for other natural language processing problems, such as word sense disambiguation, syntactic parsing, and so on.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*, pages 152–164.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL 2006*, Sydney, Australia, July.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of NIPS-2001*.

Daniel Gildea and Martha Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL-2002*, pages 239–246.

Thorsten Joachims. 2002. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow statistic parsing. In *Proceedings of ACL-2004*, pages 335–342.

Alessandro Moschitti. 2006. Syntactic kernels for natural language learning: the semantic role labeling case. In *Proceedings of the HHLT-NAACL-2006*, June.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of IJCAI-2005*.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley.

Min Zhang, Wanxiang Che, Aiti Aw, Chew Lim Tan, Guodong Zhou, Ting Liu, and Sheng Li. 2007. A grammar-driven convolution tree kernel for semantic role classification. In *Proceedings of ACL-2007*, pages 200–207.

SYNGRAPH: A Flexible Matching Method based on Synonymous Expression Extraction from an Ordinary Dictionary and a Web Corpus

Tomohide Shibata†, **Michitaka Odani**†, **Jun Harashima**†,
Takashi Oonishi††, and **Sadao Kurohashi**†

†Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
††NEC Corporation, 1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan

{shibata,odani,harashima,kuro}@nlp.kuee.kyoto-u.ac.jp
t-onishi@bq.jp.nec.com

Abstract

This paper proposes a flexible matching method that can assimilate the expressive divergence. First, broad-coverage synonymous expressions are automatically extracted from an ordinary dictionary, and among them, those whose distributional similarity in a Web corpus is high are used for the flexible matching. Then, to overcome the combinatorial explosion problem in the combination of expressive divergence, an ID is assigned to each synonymous group, and SYNGRAPH data structure is introduced to pack the expressive divergence. We confirmed the effectiveness of our method on experiments of machine translation and information retrieval.

1 Introduction

In natural language, many expressions have almost the same meaning, which brings great difficulty to many NLP tasks, such as machine translation (MT), information retrieval (IR), and question answering (QA). For example, suppose an input sentence (1) is given to a Japanese-English example-based machine translation system.

(1) *hotel ni ichiban chikai eki wa doko-desuka*
hotel best near station where is

Even if a very similar translation example (TE) "(2-a) ↔ (2-b)" exists in the TEs, a simple exact matching method cannot utilize this example for the translation.

(2) a. *ryokan no moyori no eki wa*
Japanese hotel nearest station
doko-desuka
where is
b. Where's the nearest station to the hotel?

How to handle these synonymous expressions has become one of the important research topics in NLP.

This paper presents a flexible matching method, which can assimilate the expressive divergence, to solve this problem. This method has the following two features:

1. Synonymy relations and hypernym-hyponym relations are automatically extracted from an ordinary dictionary and a Web corpus.
2. Extracted synonymous expressions are effectively handled by SYNGRAPH data structure, which can pack the expressive divergence.

An ordinary dictionary is a knowledge source to provide synonym and hypernym-hyponym relations (Nakamura and Nagao, 1988; Tsurumaru et al., 1986). A problem in using synonymous expressions extracted from a dictionary is that some of them are not appropriate since they are rarely used. For example, a synonym pair "*suidou*"[1] = "*kaikyou*(strait)" is extracted.

Recently, some work has been done on corpus-based paraphrase extraction (Lin and Pantel, 2001; Barzilay and Lee, 2003). The basic idea of their methods is that two words with similar meanings are used in similar contexts. Although their methods can obtain broad-coverage paraphrases, the obtained paraphrases are not accurate enough to be utilized

[1] This word usually means "water supply".

for achieving precise matching since they contain synonyms, near-synonyms, coordinate terms, hypernyms, and inappropriate synonymous expressions.

Our approach makes the best use of an ordinary dictionary and a Web corpus to extract broad-coverage and precise synonym and hypernym-hyponym expressions. First, synonymous expressions are extracted from a dictionary. Then, the distributional similarity of a pair of them is calculated using a Web corpus. Among extracted synonymous expressions, those whose similarity is high are used for the flexible matching. By utilizing only synonymous expressions extracted from a dictionary whose distributional similarity is high, we can exclude synonymous expressions extracted from a dictionary that are rarely used, and the pair of words whose distributional similarity is high that is not actually a synonymous expression (is not listed in a dictionary).

Another point of our method is to introduce SYNGRAPH data structure. So far, the effectiveness of handling expressive divergence has been shown for IR using a thesaurus-based query expansion (Voorhees, 1994; Jacquemin et al., 1997). However, their methods are based on a bag-of-words approach and thus does not pay attention to sentence-level synonymy with syntactic structure. MT requires such precise handling of synonymy, and advanced IR and QA also need it. To handle sentence-level synonymy precisely, we have to consider the combination of expressive divergence, which may cause combinatorial explosion. To overcome this problem, an ID is assigned to each synonymous group, and then SYNGRAPH data structure is introduced to pack expressive divergence.

2 Synonymy Database

This section describes a method for constructing a synonymy database. First, synonym/hypernym relations are automatically extracted from an ordinary dictionary, and the distributional similarity of a pair of synonymous expressions is calculated using a Web corpus. Then, the extracted synonymous expressions whose similarity is high are used for the flexible matching.

2.1 Synonym/hypernym Extraction from an Ordinary Dictionary

Although there were some attempts to extract synonymous expressions from a dictionary (Nakamura and Nagao, 1988; Tsurumaru et al., 1986), they extracted only hypernym-hyponym relations from the limited entries. In contrast, our method extracts not only hypernym-hyponym relations, but also basic synonym relations, predicate synonyms, adverbial synonyms and synonym relations between a word and a phrase.

The last word of the first definition sentence is usually the hypernym of an entry word. Some definition sentences in a Japanese dictionary are shown below (the left word of ":" is an entry word, the right sentence is a definition, and words in bold font is the extracted words):

yushoku (dinner) : *yugata* (evening) *no* (of) ***shokuji* (meal)**.

jushin (barycenter) : *omosa* (weight) *ga* (is) *tsuriatte* (balance) *tyushin* (center) *tonaru* (become) ***ten* (spot)**.

For example, the last word *shokuji* (meal) can be extracted as the hypernym of *yushoku* (dinner). In some cases, however, a word other than the last word can be a hypernym or synonym. These cases can be detected by sentence-final patterns as follows (the underlined expressions represent the patterns):

Hypernyms

dosei (Saturn) : ***wakusei* (planet)** *no* (of) *hitotsu* (one).

tobi (kite) : ***taka* (hawk)** *no* (of) *issyu* (kind).

Synonyms / Synonymous Phrases

ice : ***ice cream*** *no* (of) *ryaku* (abbreviation).

mottomo (most) : ***ichiban* (best)**. (∗ one word definition)

moyori (nearest) : ***ichiban* (best) *chikai* (near)** *tokoro* (place)[2]. (∗ less than three phrases)

2.2 Calculating the Distributional Similarity using a Web Corpus

The similarity between a pair of synonymous expressions is calculated based on *distributional similarity* (J.R.Firth, 1957; Harris, 1968) using the Web corpus collected by (Kawahara and Kurohashi, 2006). The similarity between two predicates is defined to be one between the patterns of case examples of each predicate (Kawahara and Kurohashi, 2001). The similarity between two nouns are defined

[2]If the last word of a sentence is a highly general term such as *koto* (thing) and *tokoro* (place), it is removed from the synonymous expression.

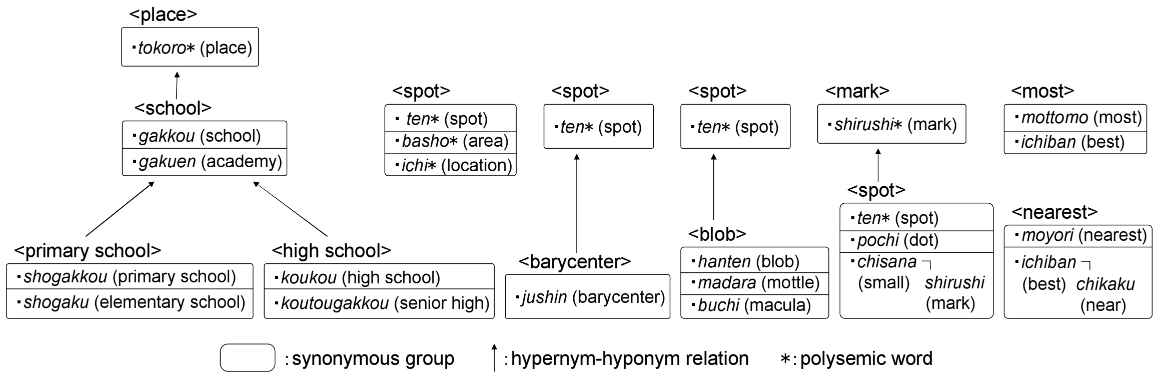


Figure 1: An example of synonymy database.

as the ratio of the overlapped co-occurrence words using the Simpson coefficient. The Simpson coefficient is computed as $\frac{|T(w_1) \wedge T(w_2)|}{min(|T(w_1)|,|T(w_2)|)}$, where $T(w)$ is the set of co-occurrence words of word w.

2.3 Integrating the Distributional Similarity into the Synonymous Expressions

Synonymous expressions can be extracted from a dictionary as described in Section 2.1. However, some extracted synonyms/hypernyms are not appropriate since they are rarely used. Especially, in the case of that a word has multiple senses, the synonym/hypernym extracted from the second or later definition might cause the inappropriate matching. For example, since "*suidou*" has two senses, the two synonym pairs, "*suidou*" = "*jyosuidou*(water supply)" and "*suidou*" = "*kaikyou*(strait)", are extracted. The second sense is rarely used, and thus if the synonymy pair extracted from the second definition is used as a synonym relation, an inappropriate matching through this synonym might be caused. Therefore, only the pairs of synonyms/hypernyms whose distributional similarity calculated in Section 2.2 is high are utilized for the flexible matching.

The similarity threshold is set to 0.4 for synonyms and to 0.3 for hypernyms. For example, since the similarity between "*suidou*" and "*kaikyou*" is 0.298, this synonym is not utilized.

2.4 Synonymy Database Construction

With the extracted binomial relations, a synonymy database can be constructed. Here, polysemic words should be treated carefully[3]. When the relations A=B and B=C are extracted, and B is not polysemic, they can be merged into A=B=C. However, if B is polysemic, the synonym relations are not merged through a polysemic word. In the same way, as for hypernym-hyponym relations, A → B and B → C, and A → B and C → B are not merged if B is polysemic. By merging binomial synonym relations with the exception of polysemic words, synonymous groups are constructed first. They are given IDs, hereafter called SYNID[4]. Then, hypernym-hyponym relations are established between synonymous groups. We call this resulting data as synonymy database. Figure 1 shows examples of synonymous groups in the synonymy database. In this paper, SYNID is denoted by using English gloss word, surrounded by " ⟨ ⟩ ".

3 SYNGRAPH

3.1 SYNGRAPH Data Structure

SYNGRAPH data structure is an acyclic directed graph, and the basis of SYNGRAPH is the dependency structure of an original sentence (in this paper, a robust parser (Kurohashi and Nagao, 1994) is always employed). In the dependency structure, each node consists of one content word and zero or more function words, which is called a *basic node* hereafter. If the content word of a basic node belongs to a synonymous group, a new node with the SYNID is attached to it, and it is called a *SYN node* hereafter. For example, in Figure 2, the shaded nodes are basic nodes and the other nodes are SYN nodes[5].

Then, if the expression conjoining two or more

[3]If a word has two or more definition items in the dictionary, the word can be regarded as polysemic.

[4]Spelling variations such as use of Hiragana, Katakana or Kanji are handled by the morphological analyzer JUMAN (Kurohashi et al., 1994).

[5]The reason why we distinguish basic nodes from SYN nodes is to give priority to exact matching over synonymous matching.

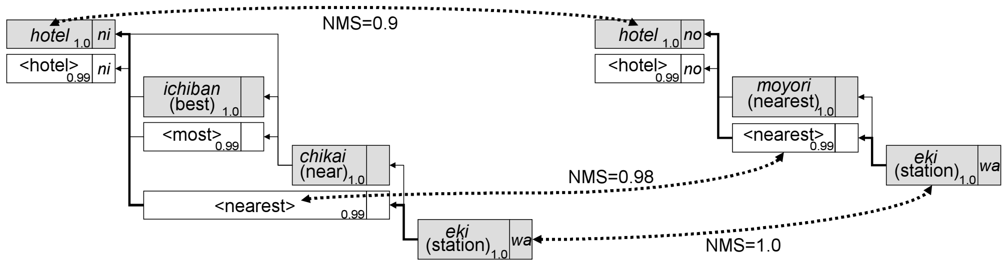


Figure 2: SYNGRAPH matching.

nodes corresponds to one synonymous group, a SYN node is added there. In Figure 2, ⟨nearest⟩ is such a SYN node. Furthermore, if one SYN node has a hyper synonymous group in the synonymy database, the SYN node with the hyper SYNID is also added.

In this SYNGRAPH data structure, each node has a score, NS (Node Score), which reflects how much the expression of the node is shifted from the original expression. We explain how to calculate NSs later.

3.2 SYNGRAPH Matching

Two SYNGRAPHs match if and only if

- all the nodes in one SYNGRAPH can be matched to the nodes in the other one,
- the matched nodes in two SYNGRAPHs have the same dependency structure, and
- the nodes can cover the original sentences.

An example of SYNGRAPH matching is illustrated in Figure 2. When two SYNGRAPHs match each other, their matching score is calculated as follows. First, the matching score of the matching two nodes, NMS (Node Match Score) is calculated with their node scores, NS_1 and NS_2,

$$NMS = NS_1 \times NS_2 \times FI_Penalty,$$

where we define FI_Penalty (Function word Inconsistency Penalty) is 0.9 when their function words are not the same, and 1.0 otherwise.

Then, the matching score of two SYNGRAPHs, SMS (SYNGRAPH Match Score) is defined as the average of NMSs weighted by the number of basic nodes,

$$SMS = \frac{\sum(\text{\# of basic nodes} \times NMS)}{\sum \text{\# of basic nodes}}.$$

In an example shown in Figure 2, the NMS of the left-hand side *hotel* node and the right-hand side *hotel* node is 0.9 (= 1.0 × 1.0 × 0.9). The NMS of the left-hand side ⟨nearest⟩ node and the right-hand side ⟨nearest⟩ node is 0.98 (= 0.99 × 0.99 × 1.0). Then, the SMS becomes $\frac{0.9\times2+0.98\times3+1.0\times2}{2+3+2} = 0.96$.

3.3 SYNGRAPH Transformation of Synonymy Database

The synonymy database is transformed into SYNGRAPHs, where SYNGRAPH matching is iteratively applied to interpret the mutual relationships in the synonymy database, as follows:

Step 1: Each expression in each synonymous group is parsed and transformed into a fundamental SYNGRAPH.

Step 2: SYNGRAPH matching is applied to check whether a sub-tree of one expression is matched with any other whole expressions. If there is a match, a new node with the SYNID of the whole matched expression is assigned to the partially matched nodes group. Furthermore, if the SYNID has a hyper synonymous group, another new node with the hypernym SYNID is also assigned. This checking process starts from small parts to larger parts.

We define the NS of the newly assigned SYN node as the SMS multiplied by a relation penalty. Here, we define the synonymy relation penalty as 0.99 and the hypernym relation penalty as 0.7. For instance, the NS of ⟨underwater⟩ node is 0.99 and that of ⟨inside⟩ node is 0.7.

Step 3: Repeat Step 2, until no more new SYN node can be assigned to any expressions. In the case of Figure 3 example, the new SYN node, ⟨diving⟩ is given to "*suityu* (underwater) *ni* (to) *moguru* (dive)" of ⟨diving(sport)⟩ at the second iteration.

4 Flexible Matching using SYNGRAPH

We use example-based machine translation (EBMT) as an example to explain how our flexible matching method works (Figure 4). EBMT generates a translation by combining partially matching TEs with an input[6]. We use flexible matching to fully exploit the TEs.

[6]How to select the best TEs and combine the selected TEs for generating a translation is omitted in this paper.

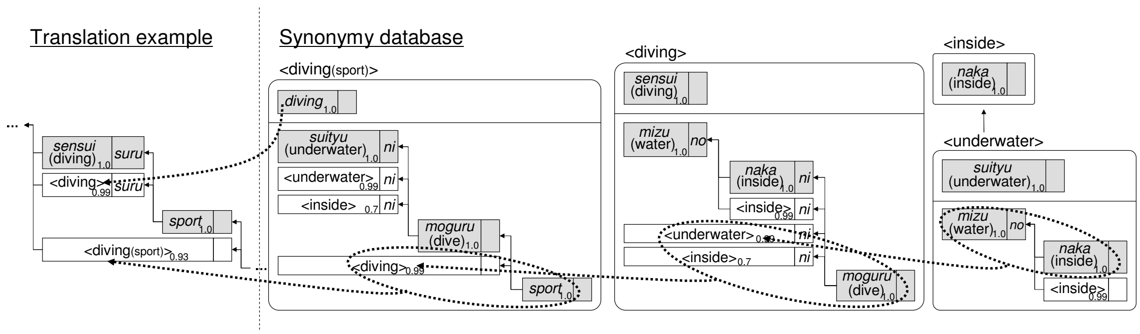


Figure 3: SYNGRAPH transformation of synonymy database.

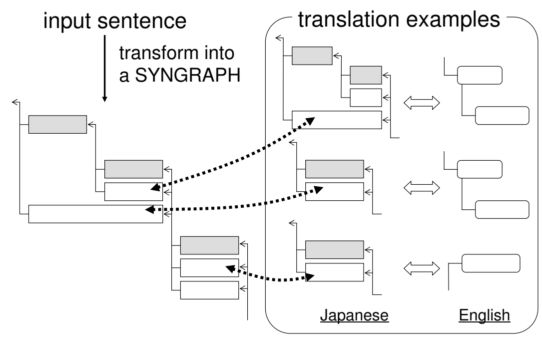


Figure 4: Flexible matching using SYNGRAPH in EBMT.

First, TEs are transformed into SYNGRAPHs by SYNGRAPH matching with SYNGRAPHs of the synonymy database. Since the synonymy database has been transformed into SYNGRAPHs, we do not need to care the combinations of synonymous expressions any more. In the example shown in Figure 3, "*sensui* (diving) *suru* (do) *sport*" in the TE is given ⟨diving(sport)⟩ node just by looking at SYNGRAPHs in ⟨diving(sport)⟩ synonymous group.

Then, an input sentence is transformed into a SYNGRAPH by SYNGRAPH matching, and then the SYNGRAPH matching is applied between all the sub trees of the input SYNGRAPH and SYNGRAPHs of TEs to retrieve the partially matching TEs.

5 Experiments and Discussion

5.1 Evaluation on Machine Translation Task

To see the effectiveness of the our proposed method, we conducted our evaluations on a MT task using Japanese-English translation training corpus (20,000 sentence pairs) and 506 test sentences of IWSLT'05[7]. As an evaluation measure, NIST and BLEU were used based on 16 reference English sentences for each test sentence.

[7] http://www.is.cs.cmu.edu/iwslt2005/.

Table 1: Size of synonymy database.

# of synonymous group	5,046
# of hypernym-hyponym relation	18,590

The synonymy database used in the experiments was automatically extracted from the REIKAI-SHOGAKU dictionary (a dictionary for children), which consists of about 30,000 entries. Table 1 shows the size of the constructed synonymy database.

As a base translation system, we used an EBMT system developed by (Kurohashi et al., 2005). Table 2 shows the experimental results. "None" means the baseline system without using the synonymy database. "Synonym" is the system using only synonymous relations, and it performed best and achieved 1.2% improvement for NIST and 0.8% improvement for BLEU over the baseline. These differences are statistically significant ($p < 0.05$). Some TEs that can be retrieved by our flexible matching are shown below:

- **input:** *fujin* (lady) *you* (for) *toile* (toilet) ↔ **TE:** *josei* (woman) *you* (for) *toile* (toilet)
- **input:** *kantan-ni ieba* (in short) ↔ **TE:** *tsumari* (in other words)

On the other hand, if the system also uses hypernym-hyponym relation ("Synonym Hypernym"), the score goes down. It proves that hypernym examples are not necessarily good for translation. For example, for a translation of *depato* (department store), its hypernym "*mise*(store)" was used, and it lowered the score.

Major errors are caused by the deficiency of word sense disambiguation. When a polysemic word occurs in a sentence, multiple SYNIDs are attached to the word, and thus, the incorrect matching might be occurred. Incorporation of unsupervised word-

Table 2: Evaluation results on MT task.

Synonymy DB	NIST	BLEU
None	8.023	0.375
Synonym	8.121	0.378
Synonym Hypernym	8.010	0.374

Table 3: Evaluation results on IR task.

Method	Synonymy DB	R-prec
Best IREX system	–	0.493
BM25	–	0.474
Our method	None	0.492
	Synonym	0.509
	Synonym Hypernym	0.514

sense-disambiguation of words in dictionary definitions and matching sentences is one of our future research targets.

5.2 Evaluation on Information Retrieval Task

To demonstrate the effectiveness of our method in other NLP tasks, we also evaluated it in IR. More concretely, we extended word-based importance weighting of Okapi BM25 (Robertson et al., 1994) to SYN node-based weighting. We used the data set of IR evaluation workshop IREX, which contains 30 queries and their corresponding relevant documents in 2-year volume of newspaper articles[8]. Table 3 shows the experimental results, which are evaluated with R-precision. The baseline system is our implementation of OKAPI BM25. Differently from the MT task, the system using both synonym and hypernym-hyponym relations performed best, and its improvement over the baseline was 7.8% relative. This difference is statistically significant ($p < 0.05$). This result shows the wide applicability of our flexible matching method for NLP tasks. Some examples that can be retrieved by our flexible matching are shown below:

- **query:** *gakkou-ni* (school) *computer-wo* (computer) *dounyuu* (introduce) ↔ **document:** *shou-gakkou-ni* (elementary school) *pasokon-wo* (personal computer) *dounyuu* (introduce)

6 Conclusion

This paper proposed a flexible matching method by extracting synonymous expressions from an ordinary dictionary and a Web corpus, and introducing SYNGRAPH data structure. We confirmed the effectiveness of our method on experiments of machine translation and information retrieval.

Our future research targets are to incorporate word sense disambiguation to our framework, and to extend SYNGRAPH matching to more structural paraphrases.

[8]http://nlp.cs.nyu.edu/irex/.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003*, pages 16–23.

Zellig Harris. 1968. *Mathematical Structures of Language*. Wiley.

Christian Jacquemin, Judith L. Klavans, and Evelyne Tzoukermann. 1997. Expansion of multi-word terms for indexing and retrieval using morphology and syntax. In *35th Annual Meeting of the Association for Computational Linguistics*, pages 24–31.

J.R.Firth. 1957. A synopsis of linguistic theory, 1933-1957. In *Studies in Linguistic Analysis*, pages 1–32. Blackwell.

Daisuke Kawahara and Sadao Kurohashi. 2001. Japanese case frame construction by coupling the verb and its closest case component. In *Proc. of HLT 2001*, pages 204–210.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proc. of LREC-06*.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proc. of the International Workshop on Sharable Natural Language*, pages 22–28.

Sadao Kurohashi, Toshiaki Nakazawa, Kauffmann Alexis, and Daisuke Kawahara. 2005. Example-based machine translation pursuing fully structural NLP. In *Proc. of IWSLT'05*, pages 207–212.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Junichi Nakamura and Makoto Nagao. 1988. Extraction of semantic information from an ordinary english dictionary and its evaluation. In *Proc. of the 12th COLING*, pages 459–464.

S. E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. 1994. Okapi at TREC-3. In *the third Text REtrieval Conference (TREC-3)*.

Hiroaki Tsurumaru, Toru Hitaka, and Sho Yoshida. 1986. An attempt to automatic thesaurus construction from an ordinary japanese language dictionary. In *Proc. of the 11th COLING*, pages 445–447.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *SIGIR*, pages 61–69.

Annotation of Multiword Expressions in the Prague Dependency Treebank

Eduard Bejček, Pavel Straňák and Pavel Schlesinger
Institute of Formal and Applied Linguistics
Charles University, Prague, Czech Republic
{bejcek, stranak, schlesinger}@ufal.mff.cuni.cz

Abstract

In this article we want to demonstrate that annotation of multiword expressions in the Prague Dependency Treebank is a well defined task, that it is useful as well as feasible, and that we can achieve good consistency of such annotations in terms of inter-annotator agreement. We show a way to measure agreement for this type of annotation. We also argue that some automatic pre-annotation is possible and it does not damage the results.

1 Motivation

Various projects involving lexico-semantic annotation have been ongoing for many years. Among those there are the projects of word sense annotation, usually for creating training data for word sense disambiguation. However majority of these projects have only annotated very limited number of word senses (cf. Kilgarriff (1998)). Even among those that aim towards "all words" word-sense annotation, multiword expressions (MWE) are not annotated adequately (see (Mihalcea, 1998) or (Hajič et al., 2004)), because for their successful annotation a methodology allowing identification of new MWEs during annotation is required. Existing dictionaries that include MWEs concentrate only on the most frequent ones, but we argue that there are many more MWEs that can only be identified (and added to the dictionary) by annotation.

There are various projects for identification of named entities (for an overview see (Ševčíková et al., 2007)). We explain below (mainly in Section 2) why we consider named entities to be concerned with lexical meaning. At this place we just wish to recall that these projects only select some specific parts of text and provide information only for these. They do not aim for full lexico-semantic annotation of texts.

There is also another group of projects that have to tackle the problem of lexical meaning, namely treebanking projects that aim to develop a deeper layer of annotation in adition to a surface syntactic layer. This deeper layer is generally agreed to concern lexical meaning. Therefore the units of this layer cannot be words anymore, they should be *lexias*.

Lexia is defined by Filipec and Čermák (1986) as equivalent to a "monosemic lexeme" of (Filipec, 1994) or a "lexical unit" of (Cruse, 1986): *"a pair of a single sense and a basic form (plus its derived forms) with relatively stable semantic properties"*.

We work with the Prague Dependency Treebank (PDT, see Hajič (2005)), which has in addition to the morphemic and the surface syntactic layers also the tectogrammatical layer. The latter has been construed as the layer of the (literal) meaning of the sentence and thus should be composed of lexias (lexical units) and the relations between their occurrences.[1]

On the tectogrammatical layer only the autosemantic words form nodes in a tree (t-nodes). Synsemantic (function) words are represented by various attributes of t-nodes. Each t-node has a lemma: an attribute whose value is the node's basic lexical form. Currently t-nodes, and consequently their t-lemmas, are still visibly derived from the morphological division of text into tokens. This preliminary handling

[1] With a few exceptions, such as personal pronouns (that corefer to other lexias) or coordination heads.

has always been considered unsatisfactory in FGD.[2] There is a clear goal to distinguish t-lemmas through their senses, but this process has not been completed so far.

Our project aims at improving the current state of t-lemmas. Our goal is to assign each t-node a t-lemma that would correspond to a lexia, i.e. that would really distinguish the t-node's lexical meanings. To achieve this goal, in the first phase of the project, which we report on in this paper, we *identify multiword expressions and create a lexicon of the corresponding lexias*.

2 Introduction

We annotate all occurrences of MWEs (including named entities, see below) in PDT 2.0. When we speak of **multiword expressions** we mean "idiosyncratic interpretations that cross word boundaries" (Sag et al., 2002). We understand multiword expressions as *a type of lexias*. We distinguish also a special type of MWEs, for which we are mainly interested in its type, rather than individual lexias, during the annotation: **named entities (NE)**.[3] Treatment of NEs together with other MWEs is important, because syntactic functions are more or less arbitrary inside a NE (consider an address with phone numbers, etc.) and so is the assignment of semantic roles. That is why we need each NE to be combined into a single node, just like we do it with MWEs in general.

For the purpose of annotation we have built a repository of lexias corresponding to MWEs, which we call SemLex. We have built it using entries from some existing dictionaries and it is being enriched during the annotation in order to contain every lexia that was annotated. We explain this in detail in Section 4.1.

3 Current state of MWEs in PDT 2.0

During the annotation of valency that is a part of the tectogrammatical layer of PDT 2.0 the t-lemmas that correspond to lexias have been basically identified for all the verbs and some nouns and adjectives. The resulting valency lexicon is called PDT-VALLEX (Hajič et al., 2003) and we can see it as a repository of lexias based on verbs, adjectives and nouns in PDT that have valency. [4]

This is a starting point for having t-nodes corresponding to lexias. However in the current state it is not fully sufficient even for verbs, mainly because parts of MWEs are not joined into one node. Parts of frames marked as idiomatic are still represented by separate t-nodes in a tectogrammatical tree. Verbal phrasemes are also split into 2 nodes, where the nominal part is governed by the verb. Non-verbal idioms have not been annotated at all.

Below we give an example of the current state: an idiom meaning "in a blink (of an eye)" – literally "*what not-see" (Figure 1).

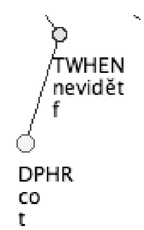


Figure 1: "Co nevidět" (in a blink)

4 Methodology

4.1 Building SemLex

Each entry we add into SemLex is considered to be a **lexia**. We have also added 9 special entries to identify NE types, so we do not need to add the expressions themselves. These types are derived from NE classification by (Ševčíková et al., 2007). Some frequent names of persons, institutions or other objects (e.g. film titles) are being added into SemLex during annotation (while keeping the information about a NE type), because this allows for their following occurrences to be pre-annotated automatically (see Section 5). For others, like addresses or bibliographic

[2]Functional Generative Description (FGD, (Sgall et al., 1986; Hajičová et al., 1998)) is a framework for systematic description of a language, that the PDT project is based upon. In FGD units of the t-layer are construed equivalently to monosemic lexemes (lexias) and are combined into dependency trees, based on syntactic valency of the lexias.

[3]NEs can in general be also single-word, but in this phase of our project we are only interested in multiword expressions, so when we say NE in this paper, we always mean multiword.

[4]It is so because in PDT-VALLEX valency is not the only criterion for distinguishing frames (=meanings). Two words with the same morphological lemma and valency frame are assigned two different frames if their meaning differs. Thus the PDT-VALLEX frames correspond to lexias.

entries, it makes but little sense, because they most probably will not reappear during the annotation.

Currently (for the first stage of lexico-semantic annotation of PDT) SemLex contains only lexias corresponding to MWEs. Its base has been composed of MWEs extracted from Czech WordNet (Smrž, 2003), Eurovoc (Eurovoc, 2007) and SČFI (Čermák et al., 1994).[5] Currently there are over 30,000 multi-word lexias in SemLex and more are being added during annotations.

The entries added by annotators must be lexias as defined above. Annotators define their "sense" informally (as much as possible) and we extract an example of usage and the basic form from the annotation automatically. The "sense" information shall be revised by a lexicographer, based on annotated occurrences.

4.2 Annotation

PDT 2.0 uses PML (Pajas and Štěpánek, 2005), which is an application of XML that utilises a stand-off annotation scheme. We have extended the PDT-PML with a new schema for so-called s-files. We use these files to store all of our annotation without altering the PDT itself. These s-files are very simple: basically each of them consists of a list of s-nodes. Each s-node corresponds to an occurrence of a MWE and it is composed of a link to the entry in SemLex and a list of identifiers of t-nodes that correspond to this s-node.

Our annotation program reads in a tectogrammatical representation (t-file) and calls TrEd (Pajas, 2007) to generate plain text. This plain text (still linked to the tectogrammatical representation) is presented to the annotator. While the annotator marks MWEs already present in SemLex or adds new MWEs into SemLex, tree representations of these MWEs extracted from underlying t-trees are added into their SemLex entries via TrEd scripts.

5 Pre-annotation

Because MWEs tend to occur repeatedly in a text, we have decided to test pre-annotation both for the speed improvement and for improving the consistency of annotations. On the assumption that *all occurrences of a MWE share the same tree structure*, while there are no restrictions on the surface word order other than those imposed by the tree structure itself we have decided to employ four types of pre-annotation:

A) External pre-annotation provided by our colleague (see Hnátková (2002)). With each MWE a set of rules is associated that limits possible forms and surface word order of parts of a MWE. This approach was devised for corpora that are not syntactically annotated.

B) Our one-time pre-annotation with those lexias from SemLex that were already used in annotation, and thus have a tree structure as a part of their entry.

C) Dynamic pre-annotation as in B, only with the SemLex entries that have been recently added by the annotator.

D) When an annotator tags an occurrence of a MWE in the text, other occurrences of this MWE in the article are identified automatically.[6]

(A) was executed once for all of the PDT. (B) is performed each time we merge lexias added by annotators into the main SemLex. We carry out this annotation in one batch for all PDT files remaining to annotate. (C) should be done for each file while it is being opened in LexemAnn GUI. (D) happens each time the annotator adds a new lexia into SemLex and uses it to annotate an occurrence in the text. In subsequent files instances of this lexia are already annotated in step (C), and later even in (B).

After the pilot annotation without pre-annotation (D) we have compared instances of the same tags and found that 10.5% of repeated lexias happened to have two different trees. After closer examination this 10.5% group is negligible because these cases are caused by ellipses, variations in lexical form such as diminutives etc., or wrong lemmatisation, rather than inconsistencies in the tree structure. These cases show us some issues of PDT 2.0, for instance:

- *jižní* × *Jižní Korea* [southern × South Korea] – wrong lemmatisation

[5] Slovník české frazeologie a idiomatiky (Dictionary of Czech Phraseology and Idiomatics)

[6] This is exactly what happens: 1) Tree structure of the selected MWE is identified via TrEd 2) The tree structure is added to the lexeme's entry in SemLex 3) All the sentences in the given file are searched for the same MWE using its tree structure (via TrEd) 4) Other occurrences returned by TrEd are tagged with this MWE's ID, but these occurrences receive an attribute "auto", which identifies them (both in the s-files and visually in the annotation tool) as annotated automatically.

- *obchodní ředitel* × *ředitelka* [managing director – man × woman] – in future these should have one t-lemma and gender should be specified by an attribute of a t-node.

We have not found any case that would show that there is such a MWE that its structure cannot be represented by a single tectogrammatical tree. 1.1% of all occurences were not connected graphs, but this happened due to errors in data and to coordination. This corroborates our assumption that (disregarding errors) all occurrences of a MWE share the same tree structure. As a result, we started storing the tree structures in the SemLex entries and employ them in pre-annotation (D). This also allows us to use pre-annotations (B) and (C), but we have decided not to use them at the moment, in order to be able to evaluate each pre-annotation step separately. Thus the following section reports on the experiments that employ pre-annotation (A) and (D).

6 Analysis of Annotations

Two annotators already started to use (and test) the tool we have developed. They both have got the same texts. The text is generated from the t-trees and presented as a plain text with pre-annotated words marked by colour labels. Annotators add their tags in the form of different colour labels and they can delete the pre-annotated tags. In this experiment data consists of approx. 120,000 tokens that correspond to 100,000 t-nodes. Both annotators have marked about 15,200 t-nodes (~15%) as parts of MWEs. annotator A has grouped them into 7,263 MWEs and annotator B into 6,888. So the average length of a MWE is 2.2 t-nodes.

The ratio of general named entities versus SemLex lexias was 52:48 for annotator A and 49:51 in case of annotator B. Annotator B used 10% more lexias than annotator A (3,279 and 3,677), while they both used almost the same number of NEs. Some comparison is in the Table 1.

type of MWE	A	B
SemLex lexias	3,677	3,279
Named Entities	3,553	3,587
- person/animal	1130	1137
- institution	842	772

Table 1: Annotated instances of significant types of MWEs

Both annotators also needed to add missing entries to the originally compiled SemLex or to edit existing entries. annotator A added 722 entries while the annotator B added 861. They modified 796 and 809 existing entries, respectively.

6.1 Inter-anntator Agreement

In this section our primary goal is to assess whether with our current methodology we produce reliable annotation of MWEs. To that end we measure the amount of inter-annotator agreement that is above chance. There are, however, a few sources of complications in measuring this agreement:

- Each tag of a MWE identifies a subtree of a tectogrammatical tree (represented on the surface by a set of marked words). This allows for partial agreement of tags at the beginning, at the end, but also in the middle of a surface interval (in a sentence).
- A disagreement of the annotators on the tag is still an agreement on the fact that this t-node is a part of a MWE and thus should be tagged. This means we have to allow for partial agreement on a tag.
- There is not any clear upper bound as to how many (and how long) MWEs are there in texts.
- There is not a clear and simple way to estimate the amount of the agreement by chance, because it must include the partial agreements mentioned above.

Since we want to keep our agreement calculation as simple as possible but we also need to take into account the problems above, we have decided to start from π as defined in (Artstein and Poesio, 2007) and to make a few adjustments to allow for types of partial agreement and estimated maximal agreement.

Because we do not know how many MWEs there are in our texts, *we need to calculate the agreement over all t-nodes*, rather than the t-nodes that "should be annotated". This also means, that the theoretical maximal agreement (upper bound) U, cannot be 1. If it was 1, it would be saying that all nodes are part of a MWE.

Since we know that $U < 1$ but we do not know it's exact value, we use the *estimated upper bound* $\widehat{U}$ (see Equation 1). Because we calculate $\widehat{U}$ over all t-nodes, we need to account not only for agreement on tagging a t-node, but also for agreement, that the t-node is not a part of a MWE, therefore it is not

tagged.[7]

If N is the number of all t-nodes in our data and $n_{A \cup B}$ is the number of t-nodes annotated by at least one annotator, then we estimate $\widehat{U}$ as follows:

$$\widehat{U} = \frac{n_{A \cup B}}{N} + 0.052 \cdot \frac{N - n_{A \cup B}}{N} = 0.215 \quad (1)$$

The weight 0.052 used for scoring the t-nodes that were not annotated is explained below. Because $\widehat{U}$ includes all the disagreements of the annotators, we believe that the real upper bound U lies somewhat below it and the agreement value 0.215 is not something that should (or could) be achieved. This is however based on the assumption that the data we have not yet seen have similar ratio of MWEs as the data we have used.

To account for partial agreement we divide the t-nodes into 5 classes c and assign each class a weight w as follows:

$c1$ If the annotators agree on the exact tag from SemLex, we get maximum information: $w = 1$

$c2$ If they agree, that the t-node is a part of a NE or they agree it is a part of some lexia from SemLex, but they do not agree which NE or which lexia, we estimate we get about a half of the information compared to $c1$: $w = 0.5$

$c3$ If they agree that the t-node is a part of a MWE, but disagree whether a NE or a lexia from SemLex, it is again half the information compared to $c2$, so $w = 0.25$

$c4$ If they agree that the t-node is not a part of a MWE, $w = 0.052$. This low value of w accounts for frequency of t-nodes that are not a part of a MWE, as estimated from data: Agreement on not annotating provides the same amount of information as agreement on annotating, but we have to take into account higher frequency of t-nodes that are not annotated:

$$c4 = c3 \cdot \frac{\sum annotated}{\sum not\ annotated} = 0.25 \cdot \frac{12797}{61433} \approx 0.052$$

$c5$ If the annotators do not agree whether to annotate a t-node or not, $w = 0$.

The number of t-nodes (n) and weights w per class c are given in Table 2.

[7] If we did not do this, there would be no difference between t-nodes, that were not tagged (annotators agreed they are not a part of a MWE) and the t-nodes that one annotator tagged and the other did not (i.e. they disagreed).

	Agreement				Disagreement
	Agreement on annotation			Not annotation	
	Agreement on NE / lexia				
	Full agreement				
class c	1	2	3	4	5
t-nodes n	10,527	2,365	389	83,287	3,988
weight w	1	0.5	0.25	0.052	0

Table 2: The agreement per class and the associated weights

Now that we have estimated the upper bound of agreement $\widehat{U}$ and the weights w for all t-nodes we can calculate our weighted version of π:

$$\pi_w = \frac{A_o - A_e}{\widehat{U} - A_e}$$

A_o is the observed agreement of annotators and A_e is the agreement expected by chance (which is similar to a baseline). π_w is thus a simple ratio of our observed agreement above chance and maximum agreement above chance.

Weights w come into account in calculation of A_o and A_e.

We calculate A_o by multiplying the number of t-nodes in each category c by that category's weight w, summing these 5 weighted sums and dividing this sum of all the observed agreement in the data by the total number of t-nodes: $A_o = \frac{1}{N} \sum_{c=1}^{5} n_c w_c = 0.160$.

A_e is the probability of agreement expected by chance over all t-nodes. This means it is the sum of the weighted probabilities of all the combinations of all the tags that can be obtained by a pair of annotators. Every possible combination of tags (including not tagging a t-node) falls into one of the categories c and thus gets the appropriate weight w. Calculating the value of A_e depends not only on values of w (see Table 2), but also on the fact that SemLex is composed of 9 entries for NE types and over 30,000 entries for individual lexias. Based on this we have obtained $A_e = 0.047$.

The resulting π_w is then

$$\pi_w = \frac{A_o - A_e}{\widehat{U} - A_e} = \frac{0.160 - 0.047}{0.215 - 0.047} = 0.6760$$

When we analyse the cases of disagreement and partial agreement we find that most of it has to do with SemLex lexias rather than NEs. This is mostly due to imperfectness of the dictionary and its size (annotators could not explore each of almost 30,000

of SemLex entries). Our current methodology, which relies too much on searching the SemLex, is also to blame. This should, however, improve by employing pre-annotation (B) and (C).

One more reason for disagreement consists in the fact that there are cases, for which non-trivial knowledge of the world is needed: "Jang Di Pertuan Agong Sultan Azlan Šáh, the sultan of the state of Perak, [. . .] flew back to Perak." Is "Sultan Azlan Šáh" still a part of the name or is it (or a part of it) a title?

The last important reason of disagreement is simple: both annotators identify *the same* part of text as MWE instances, but while searching the SemLex they choose different lexias as the tags. This can be rectified by:

- Removing duplicate entries from SemLex (currently there are many close identical entries originating from Eurovoc and Czech WordNet).
- Imploring improved pre-annotation B and C, as mentioned above.

7 Conclusion

We have annotated multi-word lexias and named entities in a part of PDT 2.0. We use tectogrammatical tree structures of MWEs for the automatic pre-annotation. In the analysis of inter-annotator agreement we show that a weighted measure that accounts for partial agreement as well as the estimation of maximal agreement is needed.

The resulting $\pi_w = 0.6760$ is statistically significant and should gradually improve as we clean up the annotation lexicon, more entries can be pre-annotated automatically, and further types of pre-annotation are employed.

8 Acknowledgement

This work has been supported by grants 1ET2011205-05 of Grant Agency of the Academy of Science of the Czech Republic, projects MSM0021620838 and LC536 of the Ministry of Education and 201/05/H014 of the Czech Science Foundation.

References

Ron Artstein and Massimo Poesio. 2007. Inter-coder agreement for computational linguistics. *Submitted to Computational Linguistics*.

F. Čermák, V. Červená, M. Churavý, and J. Machač. 1994. *Slovník české frazeologie a idiomatiky*. Academia.

D.A. Cruse. 1986. *Lexical Semantics*. Cambridge University Press.

Eurovoc. 2007. http://europa.eu/eurovoc/.

Josef Filipec and František Čermák. 1986. *Česká lexikologie*. Academia.

Josef Filipec. 1994. Lexicology and lexicography: Development and state of the research. In P. A. Luelsdorff, editor, *The Prague School of Structural and Functional Linguistics*, pages 163–183, Amsterdam/Philadelphia. J. Benjamins.

Jan Hajič, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová, Veronika Kolářová, and Petr Pajas. 2003. PDT-VALLEX. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of The Second Workshop on Treebanks and Linguistic Theories*, volume 9 of *Mathematical Modeling in Physics, Engineering and Cognitive Sciences*, pages 57–68, Vaxjo, Sweden. Vaxjo University Press.

Jan Hajič, Martin Holub, Marie Hučínová, Martin Pavlík, Pavel Pecina, Pavel Straňák, and Pavel Martin Šidák. 2004. Validating and improving the Czech WordNet via lexico-semantic annotation of the Prague Dependency Treebank. In *LREC 2004*, Lisbon.

Jan Hajič, 2005. *Insight into Slovak and Czech Corpus Linguistics*, chapter Complex Corpus Annotation: The Prague Dependency Treebank, pages 54–73. Veda Bratislava, Slovakia.

Eva Hajičová, Barbara H. Partee, and Petr Sgall. 1998. *Topic-focus articulation, tripartite structures, and semantic content*, volume 71 of *Studies in Linguistics and Philosophy*. Kluwer, Dordrecht.

Milena Hnátková. 2002. Značkování frazémů a idiomů v Českém národním korpusu s pomocí Slovníku české frazeologie a idiomatiky. *Slovo a slovesnost*.

A. Kilgarriff. 1998. Senseval: An exercise in evaluating word sense disambiguation programs. In *Proc. LREC*, pages 581–588, Granada.

Rada Mihalcea. 1998. Semcor semantically tagged corpus.

Petr Pajas and Jan Štěpánek. 2005. A Generic XML-Based Format for Structured Linguistic Annotation and Its Application to Prague DependencyTreebank 2.0. Technical Report TR-2005-29, ÚFAL MFF UK, Prague, Czech Rep.

Petr Pajas. 2007. TrEd. http://ufal.mff.cuni.cz/∼pajas/tred/index.html.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Third International Conference, CICLing*.

Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza. 2007. Zpracování pojmenovaných entit v českých textech (treatment of named entities in czech texts). Technical Report TR-2007-36, ÚFAL MFF UK, Prague, Czech Republic.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Academia/Reidel Publ. Comp., Praha/Dordrecht.

Pavel Smrž. 2003. Quality control for wordnet development. In Petr Sojka, Karel Pala, Pavel Smrž, Christiane Fellbaum, and Piek Vossen, editors, *Proceedings of the Second International WordNet Conference—GWC 2004*, pages 206–212. Masaryk University Brno, Czech Republic.

Learning Named Entity Hyponyms for Question Answering

Paul McNamee
JHU Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099, USA
paul.mcnamee@jhuapl.edu

Rion Snow
Stanford AI Laboratory
Stanford University
Stanford, CA 94305, USA
rion@cs.stanford.edu

Patrick Schone
Department of Defense
Fort George G. Meade, MD 20755-6000
pjschon@tycho.ncsc.mil

James Mayfield
JHU Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723-6099, USA
james.mayfield@jhuapl.edu

Abstract

Lexical mismatch is a problem that confounds automatic question answering systems. While existing lexical ontologies such as WordNet have been successfully used to match verbal synonyms (e.g., *beat* and *defeat*) and common nouns (*tennis* is-a *sport*), their coverage of proper nouns is less extensive. Question answering depends substantially on processing named entities, and thus it would be of significant benefit if lexical ontologies could be enhanced with additional hypernymic (i.e., is-a) relations that include proper nouns, such as *Edward Teach* is-a *pirate*. We demonstrate how a recently developed statistical approach to mining such relations can be tailored to identify named entity hyponyms, and how as a result, superior question answering performance can be obtained. We ranked candidate hyponyms on 75 categories of named entities and attained 53% mean average precision. On TREC QA data our method produces a 9% improvement in performance.

1 Introduction

To correctly extract answers, modern question answering systems depend on matching words between questions and retrieved passages containing answers. We are interested in learning hypernymic (i.e., is-a) relations involving named entities because we believe these can be exploited to improve a significant class of questions.

For example, consider the following questions:

- What island produces Blue Mountain coffee?
- In which game show do participants compete based on their knowledge of consumer prices?
- What villain is the nemesis of Dudley Do-Right?

Knowledge that *Jamaica* is an island, that *The Price is Right* is a game show, and that *Snidely Whiplash* is a villain, is crucial to answering these questions.

Sometimes these relations are evident in the same context as answers to questions, for example, in *"The island of Jamaica is the only producer of Blue Mountain coffee"*; however, *"Jamaica is the only producer of Blue Mountain coffee"* should be sufficient, despite the fact that Jamaica is an island is not observable from the sentence.

The dynamic nature of named entities (NEs) makes it difficult to enumerate all of their evolving properties; thus manual creation and curation of this information in a lexical resource such as WordNet (Fellbaum, 1998) is problematic. Pasca and Harabagiu discuss how insufficient coverage of named entities impairs QA (2001). They write:

> "Because WordNet was not designed as an encyclopedia, the hyponyms of concepts such as *composer* or *poet* are illustrations rather than an exhaustive list of instances. For example, only twelve composer names specialize the concept *composer* ... Consequently, the enhancement of WordNet with NE information could help QA."

The chief contribution of this study is demonstrating that an automatically mined knowledge base, which naturally contains errors as well as correctly distilled knowledge, can be used to improve QA performance. In Section 2 we discuss prior work in identifying hypernymic relations. We then explain our methods for improved NE hyponym learning and its evaluation (Section 3) and apply the relations that are discovered to enhance question answering (Section 4). Finally we discuss our results (Section 5) and present our conclusions (Section 6).

2 Hyponym Induction

We review several approaches to learning is-a relations.

2.1 Hearst Patterns

The seminal work in the field of hypernym learning was done by Hearst (1992). Her approach was to identify discriminating lexico-syntactic patterns that suggest hypernymic relations. For example, "X, such as Y", as in *"elements, such as chlorine and fluorine"*.

2.2 KnowItAll

Etzioni et al. developed a system, *KnowItAll*, that does not require training examples and is broadly applicable to a variety of classes (2005). Starting with seed examples generated from high precision generic patterns, the system identifies class-specific lexical and part-of-speech patterns and builds a Bayesian classifier for each category. *KnowItAll* was used to learn hundreds of thousands of class instances and clearly has potential for improving QA; however, it would be difficult to reproduce the approach because of information required for each class (i.e., specifying synonyms such as *town* and *village* for *city*) and because it relies on submitting a large number of queries to a web search engine.

2.3 Query Logs

Pasca and Van Durme looked at learning entity class membership for five high frequency classes (*company*, *country*, *city*, *drug*, and *painter*), using search engine query logs (2007). They reported precision at 50 instances between 0.50 and 0.82.

2.4 Dependency Patterns

Snow et al. have described an approach with several desirable properties: (1) it is weakly-supervised and only requires examples of hypernym/hyponym relations and unannotated text; (2) the method is suitable for both common and rare categories; and, (3) it achieves good performance without post filtering using the Web (2005; 2006). Their method relies on dependency parsing, a form of shallow parsing where each word modifies a single parent word.

Hypernym/hyponym word pairs where the words[1] belong to a single WordNet synset were identified and served to generate training data in the following way: making the assumption that when the two words co-occur, evidence for the is-a relation is present, sentences containing both terms were extracted from unlabeled text. The sentences were parsed and paths between the nouns in the dependency trees were calculated and used as features in a supervised classifier for hypernymy.

3 Learning Named Entity Hyponyms

The present work follows the technique described by Snow et al.; however, we tailor the approach in several ways. First, we replace the logistic regression model with a support vector machine (SVM-Light). Second, we significantly increase the size of training corpora to increase coverage. This beneficially increases the density of training and test vectors. Third, we include additional features not based on dependency parses (e.g., morphology and capitalization). Fourth, because we are specifically interested in hypernymic relations involving named entities, we use a bootstrapping phase where training data consisting primarily of common nouns are used to make predictions and we then manually extract named entity hyponyms to augment the training data. A second learner is then trained using the entity-enriched data.

3.1 Data

We rely on large amounts of text; in all our experiments we worked with a corpus from the sources given in Table 1. Sentences that presented difficulties in parsing were removed and those remaining

[1]Throughout the paper, use of the term *word* is intended to include named entities and other multiword expressions.

Table 1: Sources used for training and learning.

	Size	Sentences	Genre
TREC Disks 4,5	81 MB	0.70 M	Newswire
AQUAINT	1464 MB	12.17 M	Newswire
Wikipedia (4/04)	357 MB	3.27 M	Encyclopedia

Table 2: Characteristics of training sets.

	Pos. Pairs	Neg. Pairs	Total Features
Baseline	7975	63093	162528
+NE	9331	63093	164298
+Feat	7975	63093	162804

were parsed with MINIPAR (Lin, 1998). We extracted 17.3 million noun pairs that co-occurred in at least one sentence. All pairs were viewed as potential hyper/hyponyms.

Our three experimental conditions are summarized in Table 2. The baseline model used 71068 pairs as training data; it is comparable to the weakly-supervised hypernym classifier of Snow et al. (2005), which used only dependency parse features, although here the corpus is larger. The entity-enriched data extended the baseline training set by adding positive examples. The +Feat model uses additional features besides dependency paths.

3.2 Bootstrapping

Our synthetic data relies on hyper/hyponym pairs drawn from WordNet, which is generally rich in common nouns and lacking in proper nouns. But certain lexical and syntactic features are more likely to be predictive for NE hyponyms. For example, it is uncommon to precede a named entity with an indefinite article, and certain superlative adjectives are more likely to be used to modify classes of entities (e.g., "the *youngest* coach", "the *highest* peak"). Accordingly we wanted to enrich our training data with NE exemplars.

By manually reviewing highly ranked predictions of the baseline system, we identified 1356 additional pairs to augment the training data. This annotation took about a person-day. We then rescanned the corpus to build training vectors for these co-occurring nouns to produce the +NE model vectors.

Table 3: Features considered for +Feat model.

Feature	Comment
Hypernym contained in hyponym	*Sands Hotel* is-a *hotel*
Length in chars / words	Chars: 1-4, 5-8, 9-16, 17+ Words: 1, 2, 3, 4, 5, 6, 7+
Has preposition	*Treaty of Paris*; *Statue of Liberty*
Common suffixes	*-ation*, *-ment*, *-ology*, etc...
Figurative term	Such as *goal*, *basis*, or *problem*
Abstract category	Like *person*, *location*, *amount*
Contains digits	Usually not a good hyponym
Day of week; month of year	Indiscriminately co-occurs with many nouns.
Presence and depth in WordNet graph	Shallow hypernyms are unlikely to have entity hyponyms. Presence in WN suggests word is not an entity.
Lexname of 1st synset in WordNet	Root classes like *person*, *location*, *quantity*, and *process*.
Capitalization	Helps identify entities.
Binned document frequency	Partitioned by base 10 logs

3.3 Additional Features

The +Feat model incorporated an additional 276 binary features which are listed in Table 3. We considered other features such as the frequency of patterns on the Web, but with over 17 million noun pairs this was computationally infeasible.

3.4 Evaluation

To compare our different models we created a test set of 75 categories. The classes are diverse and include personal, corporate, geographic, political, artistic, abstract, and consumer product entities. From the top 100 responses of the different learners, a pool of candidate hyponyms was created, randomly reordered, and judged by one of the authors. To assess the quality of purported hyponyms we used average precision, a measure in ranked information retrieval evaluation, which combines precision and recall.

Table 4 gives average precision values for the three models on 15 classes of mixed difficulty[2]. Performance varies considerably based on the hypernym category, and for a given category, by classifier. *N* is the number of known correct instances found in the pool that belong to a given category.

Aggregate performance, as mean average precision, was computed over all 75 categories and is

[2]These are not the highest performing classes

Table 4: Average precision on 15 categories.

	N	Baseline	+NE	+Feat
chemical element	78	0.9096	0.9781	0.8057
african country	48	0.8581	0.8521	0.4294
prep school	26	0.6990	0.7098	0.7924
oil company	132	0.6406	0.6342	0.7808
boxer	109	0.6249	0.6487	0.6773
sculptor	95	0.6108	0.6375	0.8634
cartoonist	58	0.5988	0.6109	0.7097
volcano	119	0.5687	0.5516	0.7722
horse race	23	0.4837	0.4962	0.7322
musical	80	0.4827	0.4270	0.3690
astronaut	114	0.4723	0.5912	0.5738
word processor	26	0.4437	0.4426	0.6207
chief justice	115	0.4029	0.4630	0.5955
perfume	43	0.2482	0.2400	0.5231
pirate	10	0.1885	0.3070	0.2282

Table 5: Mean average precision over 75 categories.

	Baseline	+NE	+Feat
MAP	0.4801	0.5001 (+4.2%)	0.5320 (+10.8%)

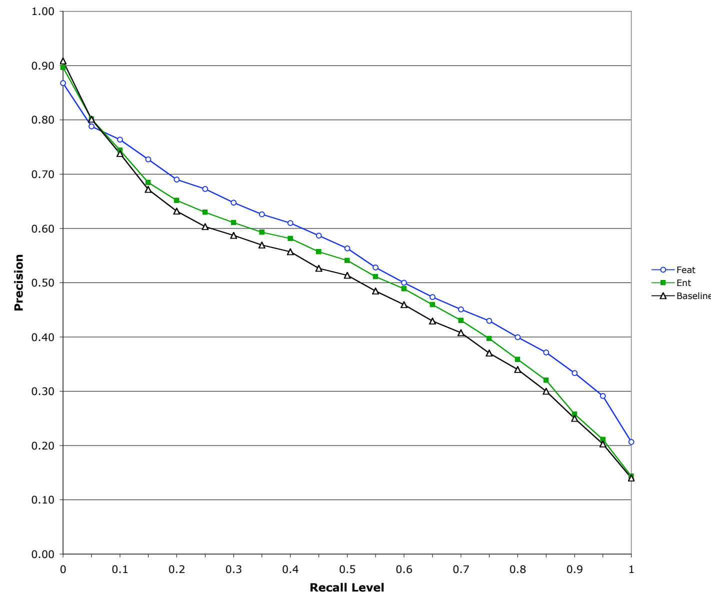


Figure 1: Precision-recall graph for three classifiers.

given in Table 5. Both the +NE and +Feat models yielded improvements that were statistically significant at a 99% confidence level. The +Feat model gained 11% over the baseline condition. The maximum F-score for +Feat is 0.55 at 70% recall.

Mean average precision emphasizes precision at low ranks, so to capture the error characteristics at multiple operating points we present a precision-recall graph in Figure 1. The +NE and +Feat models both attain superior performance at all but the lowest recall levels. For question answering this is important because it is not known which entities will be the focus of a question, so the ability to deeply mine various entity classes is important.

Table 6 lists top responses for four categories.

3.5 Discussion

53% mean average precision seems good, but is it good enough? For automated taxonomy construction precision of extracted hyponyms is critically important; however, because we want to improve question answering we prefer high recall and can tolerate some mistakes. This is because only a small set of passages that are likely to contain an answer are examined in detail, and only from this subset of passages do we need to reason about potential hyponyms. In the next section we describe an experiment which confirms that our learned entity hyponyms are beneficial.

4 QA Experiments

4.1 QACTIS

To evaluate the usefulness of our learned NE hyponyms for question answering, we used the QACTIS system (Schone et al., 2005). QACTIS was fielded at the 2004-2006 TREC QA evaluations and placed fifth at the 2005 workshop. We worked with a version of the software from July 2005.

QACTIS uses WordNet to improve matching of question and document words, and a resource, the Semantic Forest Dictionary (SFD), which contains many hypernym/hyponym pairs. The SFD was populated through both automatic and manual means (Schone et al., 2005), and was updated based on questions asked in TREC evaluations through 2004.

4.2 Experimental Setup

We used factoid questions from the TREC 2005-2006 QA evaluations (Voorhees and Dang, 2005) and measured performance with mean reciprocal rank (MRR) and percent correct at rank 1.

All runs made use of WordNet 2.0, and we examined several other sources of hypernym knowl-

Table 6: Top responses for four categories using the +Feat model. Starred entries were judged incorrect.

	Sculptor	Horse Race	Astronaut	Perfume
1	Evelyn Beatrice Longman	Tevis Cup	Mark L Polansky	* Avishag
2	Nancy Schon	Kenilworth Park Gold Cup	Richard O Covey	Ptisenbon
3	Phidias	Cox Plate	George D Nelson	Poeme
4	Stanley Brandon Kearl	Grosser Bugatti Preis	Guion Bluford Jr	Parfums International
5	Andy Galsworthy	Melbourne Cup	Stephen S Oswald	Topper Schroeder
6	Alexander Collin	* Great Budda Hall	Eileen Collins	* Baccarin
7	Rachel Feinstein	Travers Stakes	Leopold Eyharts	Pink Lady
8	Zurab K Tsereteli	English Derby	Daniel M Tani	Blue Waltz
9	Bertel Thorvaldsen	* Contrade	Ronald Grabe	WCW Nitro
10	Cildo Meireles	Palio	* Frank Poole	Jicky

Table 7: Additional knowledge sources by size.

	Classes	Class Instances
Baseline	76	11,066
SFD	1,140	75,647
SWN	7,327	458,370
+Feat	44,703	1,868,393

edge. The baseline condition added a small subset of the Semantic Forest Dictionary consisting of 76 classes seen in earlier TREC test sets (e.g., nationalities, occupations, presidents). We also tested: (1) the full SFD; (2) a database from the Stanford Wordnet (SWN) project (Snow et al., 2006); and, (3) the +Feat model discussed in Section 3. The number of classes and entries of each is given in Table 7.

4.3 Results

We observed that each source of knowledge benefitted questions that were incorrectly answered in the baseline condition. Examples include learning a meteorite (Q84.1), a university (Q93.3), a chief operating officer (Q108.3), a political party (Q183.3), a pyramid (Q186.4), and a movie (Q211.5).

In Table 8 we compare performance on questions from the 2005 and 2006 test sets. We assessed performance primarily on test questions that were deemed likely to benefit from hyponym knowledge – questions that had a readily discernible category (e.g., *"What film ..."*, *"In what country ..."*) – but we also give results on the entire test set.

The WordNet-only run suffers a large decrease compared to the baseline. This is expected because WordNet lacks coverage of entities and the baseline condition specifically populates common categories of entities that have been observed in prior TREC evaluations. Nonetheless, WordNet is useful to the system because it addresses lexical mismatch that does not involve entities.

The full SFD, the SWN, and the +Feat model achieved 17%, 2%, and 9% improvements in answer correctness, respectively. While no model had exposure to the 2005-2006 TREC questions, the SFD database was manually updated based on training on the TREC-8 through TREC-2004 data sets. It approximates an upper bound on gains attributable to addition of hyponym knowledge: it has an unfair advantage over the other models because recent question sets use similar categories to those in earlier TRECs. Our +Feat model, which has no bias towards TREC questions, realizes larger gains than the SWN. This is probably at least in part because it produced a more diverse set of classes and a significantly larger number of class instances. Compared to the baseline condition the +Feat model sees a 7% improvement in mean reciprocal rank and a 9% improvement in correct first answers; both results represent a doubling of performance compared to the use of WordNet alone. We believe that these results illustrate clear improvement attributable to automatically learned hyponyms.

The rightmost columns in Table 8 reveal that the magnitude of improvements, when measured over all questions, is less. But the drop off is consistent with the fact that only one third of questions have clear need for entity knowledge.

5 Discussion

Although there is a significant body of work in automated ontology construction, few researchers have examined the relationship between their methods

Table 8: QA Performance on TREC 2005 & 2006 Data

	Hyponym-Relevant Subset (242)		**All Questions (734)**	
	MRR	**% Correct**	**MRR**	**% Correct**
WN-alone	0.189 (-45.6%)	12.8 (-51.6%)	0.243 (-29.0%)	18.26 (-30.9%)
Baseline	0.348	26.4	0.342	26.4
SFD	0.405 (+16.5%)	31.0 (+17.2%)	0.362 (+5.6%)	27.9 (+5.7%)
SWN	0.351 (+1.0%)	26.9 (+1.6%)	0.343 (+0.3%)	26.6 (+0.5%)
Feat	0.373 (+7.4%)	28.9 (+9.4%)	0.351 (+2.5%)	27.3 (+3.1%)

for knowledge discovery and improved question-answering performance. One notable study was conducted by Mann (2002). Our work differs in two ways: (1) his method for identifying hyponyms was based on a single syntactic pattern, and (2) he looked at a comparatively simple task – given a question and one answer sentence containing the answer, extract the correct named entity answer.

Other attempts to deal with lexical mismatch in automated QA include rescoring based on syntactic variation (Cui et al., 2005) and identification of verbal paraphrases (Lin and Pantel, 2001).

The main contribution of this paper is showing that large-scale, weakly-supervised hyponym learning is capable of producing improvements in an end-to-end QA system. In contrast, previous studies have generally presented algorithmic advances and showcased sample results, but failed to demonstrate gains in a realistic application. While the hypothesis that discovering is-a relations for entities would improve factoid QA is intuitive, we believe these experiments are important because they show that automatically distilled knowledge, even when containing errors that would not be introduced by human ontologists, is effective in question answering systems.

6 Conclusion

We have shown that highly accurate statistical learning of named entity hyponyms is feasible and that bootstrapping and feature augmentation can significantly improve classifier accuracy. Mean average precision of 53% was attained on a set of 75 categories that included many fine-grained entity classes. We also demonstrated that mining knowledge about entities can be directly applied to question answering, and we measured the benefit on TREC QA data. On a subset of questions for which NE hyponyms are likely to help we found that learned hyponyms generated a 9% improvement in performance compared to a strong baseline.

References

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *SIGIR 2005*, pages 400–407.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana M. Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165(1):191–134.

Christine Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *ACL 1992*, pages 539–545.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems*.

Gideon S. Mann. 2002. Fine-grained proper noun ontologies for question answering. In *COLING-02 on SEMANET*, pages 1–7.

Marius Pasca and Benjamin Van Durme. 2007. What you seek is what you get: Extraction of class attributes from query logs. In *IJCAI-07*, pages 2832–2837.

Marius Pasca and Sanda M. Harabagiu. 2001. The informative role of wordnet in open-domain question answering. In *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources*.

Patrick Schone, Gary Ciany, Paul McNamee, James Mayfield, and Thomas Smith. 2005. QACTIS-based Question Answering at TREC 2005. In *Proceedings of the 14th Text REtrieval Conference*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *NIPS 17*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL 2006*, pages 801–808.

Errgrams – A Way to Improving ASR for Highly Inflected Dravidian Languages

Kamadev Bhanuprasad
Andhra University, Visakhapatnam
Andhra Pradesh 530 003
India
save.climate@gmail.com

Mats Svenson
University of Macau
Av. Padre Toms Pereira, Taipa
Macau, China
svmats@yahoo.com

Abstract

In this paper, we present results of our experiments with ASR for a highly inflected Dravidian language, Telugu. First, we propose a new metric for evaluating ASR performance for inflectional languages (Inflectional Word Error Rate IWER) which takes into account whether the incorrectly recognized word corresponds to the same lexicon lemma or not. We also present results achieved by applying a novel method – errgrams – to ASR lattice. With respect to confidence scores, the method tries to learn typical error patterns, which are then used for lattice correction, and applied just before standard lattice rescoring. Our confidence measures are based on word posteriors and were improved by applying antimodels trained on anti-examples generated by the standard N-gram language model. For Telugu language, we decreased the WER from 45.2% to 40.4% (by 4.8% absolute), and the IWER from 41.6% to 39.5% (2.1 % absolute), with respect to the baseline performance. All improvements are statistically significant using all three standard NIST significance tests for ASR.

1 Introduction

Speech recognition technologies allow computers equipped with a source of sound input, such as a microphone, to interpret human speech, for example, for transcription or as an alternative method of interacting with a machine. Using constrained grammar recognition (described below), such applications can achieve remarkably high accuracy. Research and development in speech recognition technology has continued to grow as the cost for implementing such voice-activated systems has dropped and the usefulness and efficiency of these systems has improved. Furthermore, speech recognition has enabled the automation of certain applications that are not automatable using push-button interactive voice response (IVR) systems. Speech recognition system are based on simplified stochastic models, so any aspects of the speech that may be important to recognition but are not represented in the models cannot be used to aid in recognition. An essential part of each Automatic Speech Recognition (ASR) system is Language Model (LM) (Rabiner L. and Juang BH., 1993; Huang X., 2001; Jelinek F., 1998). For languages with rich inflection, language modeling is difficult (Ircing P. et al., 2001; Rotovnik T. et al., 2007). To be able to perform Very (300K+) Large Vocabulary Continuous Speech Recognition in real (or at least acceptable) time, nowadays, it is often only possible to use 2-gram LM for the first recognition pass. Using only one word context is usually insufficient in order to achieve good results. To improve performance for off-line ASR, it is possible to rescore output lattice afterward (Chelba and Jelinek, 1999; Richardson F. et al., 1995; Finke et

al., 1999; Ircing P. and Psutka J., 2002). In this paper, we describe our method for reducing error rates, that was applied to improve ASR results for LVCSR of Dravidian languages namely the Telugu language.

2 Telugu and Dravidian languages in general

Since there have not yet been many publications on ASR for the Dravidian languages, we give here some basic information on them. Dravidian languages are spoken by more than 200 million people (Wikipedia, 2007). In phonology, Dravidian languages suffer from the lack of distinction between aspirated and unaspirated stops. While some Dravidian languages have large numbers of loan words from Sanskrit and other Indo-European languages, the words are often mispronounced by monolingual Dravidian speakers. Dravidian languages are also characterized by a three-way distinction between dental, alveoar and retroflex places of articulation as well as large numbers of liquids.

In this work, we show evidence from one particular Dravidian language Telugu. Telugu belongs to the family but with ample influences by the Indo-Arian family and is the official language of the state of Andhra Pradesh, India. It is the Dravidian language with the greatest number of speakers, the second largest spoken language in India after Hindi and one of the 22 official national languages of India.

The Telugu script is believed to descend from the Brahmi script of the Ashokan era. Merchants took the Eastern Chalukyan Script to Southeast Asia where it parented the scripts of Mon, Burmese, Thai, Khmer, C'am, Javanese and Balinese languages. Their similarities to Telugu script can be discerned even today. Its appearance is quite similar to the Kannada script, its closest cousin. Telugu script is written from left to right and consists of sequences of simple and/or complex characters. The script is largely syllabic in nature - the basic units of writing are syllables. Since the number of possible syllables is very large, syllables are composed of more basic units such as vowels (achchu or swar) and consonants (hallu or vyanjan). Consonants in consonant clusters take shapes which are very different from the shapes they take elsewhere. Consonants are presumed to be pure consonants, that is, without any vowel sound in them. However, it is traditional to write and read consonants with an implied 'a' vowel sound. When consonants combine with other vowel signs, the vowel part is indicated orthographically using signs known as vowel maatras. The shapes of vowel maatras are also very different from the shapes of the corresponding vowels. The overall pattern consists of 60 symbols, of which 16 are vowels, 3 vowel modifiers, and 41 consonants. Spaces are used between words as word separators. The sentence ends with either a single (purna virama) or a double bar (deergha virama). They also have a set of symbols for numerals, though Arabic numbers are typically used.

In Telugu, Karta (nominative case or the doer), Karma (object of the verb), and Kriya (action or the verb) follow a sequence. This is one of the several reasons why linguists classify Telugu as a Dravidian Language – this pattern is found in other Dravidian languages but not in Sanskrit. Telugu allows for polyagglutination, the unique feature of being able to add multiple suffixes to words to denote more complex features. Telugu also exhibits one of the rare features that Dravidian languages share with few others: the inclusive and exclusive we. The bifurcation of the First Person Plural pronoun (we in English) into inclusive (manamu) and exclusive (memu) versions can also be found in Tamil and Malayalam. Like all Dravidian languages, Telugu has a base (or of words which are essentially Dravidian in origin.

Telugu pronouns follow the systems for gender and respect also found in other Indian languages. The second person plural 'miru' is used in addressing someone with respect, and there are also respectful third personal pronouns pertaining to both genders. A specialty of the Telugu language, however, is that the third person non-respectful feminine is used to refer to objects, and there is no special 'neuter' gender that is used.

3 Method

3.1 Data

We have recorded a large broadcast news corpus for Telugu. All commercials and stretches of spontaneous speech were removed from the data, since we focus here on ASR for an unexplored language rather than on dealing with automatic audio segmentation and spontaneous speech recognition. Overall, we had at disposal 61.2 hours of pure transcribed speech. It yields 635k word tokens, contained in manual human transcriptions. Due to rich morphology of Dravidian languages, it represents 78k different word forms, with plenty of words appearing just once. We used ∼70% of data for training, ∼15% for development, and the remainding ∼15% for testing.

For language modeling, we used a newspaper corpus containing data from three major Telugu newspapers - Andhra Prabha, Eenadu, and Vaartha. This corpus contains 20M tokens, which corresponds to 615k different word forms.

3.2 Evaluation method

The usual metric to evaluate ASR system performance is Word Error Rate (WER). Unfortunately, as we described in Section 2, Telugu is a highly inflectional language having a really high number of different word forms. Using WER, this cause to underestimate the real system performance, since this metric does not distinguish between confusing word identities and confusing just forms of the same word (lemma). However, it is obvious that these errors do not have the same influence on the usability of automatic transcripts. Taking an example from English, recognizing who instead of whom is not that bad as confusing boom (especially when most Americans or not able to distinguish who and whom anyway).

Thus, we propose to use Inflectional Word Error Rate (IWER), which gives weight 1 to errors confusing lemmas, while only a weight 0.5 when the lemma of the incorrectly recognized word is correct, but the whole word form is not correct. Lemmas corresponding to particular word forms may be obtained using an automatic lemmatization technique.

3.3 Confidence measuring

The key problem for our method (as described below) is to perform appropriate ASR confidence measuring. Confidence measures (CMs) need to be interpreted in order to decide whether a word is probably recognized correct or incorrect. In this paper, we use a confidence measure based on posterior probability formulation. It is well known that the conventional ASR algorithm is usually formulated as a pattern classification problem using the maximum a posterior (MAP) decision rule to find the most likely sequence of words W which achieves the maximum posterior probability $p(W|X)$ given any acoustic observation X.

Obviously, the posterior probability $p(W|X)$ is a good confidence measure for the recognition decision that X is recognized as W. However, most real-world ASR systems simply ignore the term $p(X)$ during the search, since it is constant across different words W. This explains why the raw scores are not usable as confidence scores to reflect recognition reliability. Anyway, after the normalization by $p(X)$, the posterior probability $p(W|X)$ can be employed as a good confidence measure; it represents the absolute quantitative measure of the correspondence between X and W.

In real-world tasks, we have to either employ certain simplifying assumptions or adopt some approximate methods when estimating $p(X)$ in order to obtain the desired posteriors. In the first category, it includes the so-called filler-based methods which try to calculate $p(X)$ from a set of general filler or background models. These approaches are very straightforward and usually can achieve an reasonable performance in many cases. However, we rather used the so-called lattice-based methods which attempt to calculate $p(X)$, then the posterior probability $p(W|X)$ in turn, from a word lattice or graph based on the forwardbackward algorithm, such as Schaaf (Schaaf T. and Kemp T., 1997) and Wessel (Wessel F. et al., 1999) and their colleagues, among others.

Usually, a single word lattice or graph is generated by the ASR decoder for every "utterance".

Then, the posterior probability of each recognized word or the whole hypothesized stream of words can be calculated based on the word-graph from an additional post-processing stage. Since word graph is a compact and fairly accurate representation of all alternative competing hypotheses of the recognition result which usually dominate the summation when computing $p(X)$ over a variety of hypotheses, the posterior probability calculated from a word graph can approximate the true $p(W|X)$ very well.

In our approach, we extended the lattice based CM by using an *antimodel*. The idea of antimodels has already been proposed for CMs (Rahim M. et al., 1997), however, it has remained unclear what data should be used to estimate these antimodels. In our work, we simply generated anti-examples from our N-gram model. The rationale behind this is very straightforward. LM constraints are very strong in determining the final ASR hypotheses, and may sometimes undesirably wash out correct acoustic posteriors. Also, when you let your LM generate sentences, these sentences correspond well to N-gram probabilities but are definitely neither grammatically nor semantically correct. Thus, these generated sentences can be very well used as anti-examples to train the antimodel. Then, we performed forced-alignment against a random transcript to generate training data for each anti-model.

3.4 Errgrams

The main problem when applying ASR to extremely inflected languages such as Telugu, is the need to use a very large vocabulary, in order to reduce the OOV rate to an acceptable level. However, this causes problems for making the automatic transcription in a time close to the real-time. Since we cannot use such a big dictionary in these task, our first results had quite high WERs and IWERs. However, we analyzed the errors and found that some typical error patterns occur repeatedly. This fact inspired us to design and employ the following method.

First, using HTK large vocabulary speech recognizer (HTK, 2007) and a bigram LM, we generated an N-best ASR output and a scored bigram lattice. Then we statistically analyzed the errors and created so-called *errgrams*. Errgrams are pairs of bigrams, the first member of the pair is the correct bigram and the second member is the recognized bigram. For infrequent bigrams, the method is allowed to back-off to unigrams, using discounting based on common smoothing strategies (such Katz backoff), but the backoff is more penalized since unigram errgrams are much less reliable compared to common language modeling backoffs (such as backoff for training LMs for ASR). Errgrams were not only trained using 1-best ASR output, but to gain more real ASR data, we used 5-best hypothesis for training. For estimating errgram pairs, we also take into account confidence scores - the lower CM, the higher weight is given to a particular errgram example. By this approach, we may achieve better results with using vocabulary of standard size ($< 100k$), since words in "correct" parts of errgrams may include words that are not in the limited size vocabulary used for the first recognition pass. In other words, we can partially reduce the OOV problem by this approach. Note that LMs used for lattice rescoring include all such words originally missing in the baseline LM but appearing in errgrams.

The errgrams trained in the above described way, are then applied in the following way during the decoding phase:

1. Using a bigram model, generate an ASR lattice
2. Walk through the lattice and look for bigrams (or unigrams) having a low CM
3. If for such a low CM n-gram we have a corresponding errgram with $p >$ *Threshold*, subtract majority (particular percent is optimized on held-out data) of the probability mass and add it to the "correct" part of the errgram
4. Perform standard lattice rescoring using four-gram LMs

4 Results

Table 1 shows the comparison of WERs and IWERs for Telugu LVCSR achieved by various post-

processing methods. The baseline was achieved using just the first bigram pass. Then, we report results obtained by standard lattice rescoring method, using a fourgram LM, as well as results which were achieved by applying errgram method prior to lattice rescoring. The improvement was achieved by applying the errgram correction method. We decreased the WER from 45.2% to 40.4% (by 4.8% absolute), and the IWER from 41.6% to 39.5% (2.1%absolute), with respect to the baseline performance. As you can see, WER dropped more than the IWER did. This may be understood as that the errgrams help more in correcting errors in grammatical agreement, i.e. when the word forms differs but the lemmas are recognized correctly. The improvement from baseline to the best system is significant at $p < 0.01$ using all three NIST significance tests, while the improvement from standard lattice rescoring system is significant at $p < 0.05$, using the same statistical tests.

5 Summary, conlusions, and future work

In this paper, we have presented a very LVCSR for the highly inflected Dravidian language, namely Telugu. A new metric for evaluating ASR performance for inflectional languages, Inflectional Word Error Rate – IWER, taking into account whether incorrectly recognized words correspond to the same lemma or not, was proposed to be used together with the standard WER. We also present results achieved by applying a novel method errgrams to ASR lattice. With respect to confidence scores, the method tries to learn typical error patterns, which are then used for lattice correction, and applied just before standard lattice rescoring. By this approach, we may achieve better results with using vocabulary of standard size ($< 100k$).

The improvement was achieved by applying the errgram correction method. We decreased the WER from 45.2% to 40.4% (by 4.8% absolute), and the IWER from 41.6% to 39.5% (2.1% absolute), with respect to the baseline performance. All improvements are statistically significant using all three standard NIST significance tests for ASR.

Since this method is completely new, there is a lot of space for potential improvements. In our future work, we would definitely like to focus on improving the errgram estimation and smoothing techniques, as well as to finding the best approach for lattice rescoring. Moreover, we would like to apply our idea to other inflected languages, such as Arabic, Slovenian, Estonian or Russian. We also hope that our Telugu language will draw more attention of ASR engineers.

In the near future, we plan to largely extend our research on automic processing of spoken Telugu, especially move toward processing of spontaneous speech. Currently, we are preparing new large database of conversational speech which will be annotated with MDE-style structural metadata symbols (Strassel et al., 2005), reflecting spontaneous speech events such as fillers and edit dysfluencies. We are looking forward to test our methods on this challenging data, and compare the results with the broadcast news data used in this work.

6 Aknowledgements

The authors thank the linguists from the Macau University for kindly discussing the inflected language ASR issues. The scientific work was kindly cofunded by Anand Foundation, Raandhanpuur Foundation and Scientific Agency of Macau. All views presented in this paper are only the views of authors, and do not necessarily reflect the view by funding agencies.

References

Rabiner L., Juang BH 1993. Fundamentals of Speech Recognition, Prentice Hall PTR; 1. edition, 0130151572

Huang, X. 2001. Spoken Language Processing: A Guide to Theory, Algorithm and System Development, Prentice Hall PTR; 1st edition, 0130226165

Jelinek, F. 1998. Statistical Methods for Speech Recognition (Language, Speech, and Communication), The MIT Press , 0262100665

Ircing, P., Psutka, J.,Krbec P., Hajic J., Khudanpur S., Jelinek F., Byrne W: 2001. On Large Vocabulary

Table 1: WER[%] and Inflectional WER(IWER)[%] for LVCSR Telugu ASR using various methods

	WER [%]	IWER [%]
1st bigram pass	45.2	41.6
fourgram lattice rescoring	42.3	40.2
errgrams+lattice rescoring	40.4	39.5

Continuous Speech Recognition of Highly Inflectional Language, Eurospeech Aalborg

T. Rotovnik, M.S. Maucec, Z. Kacix. 2007. Large vocabulary continuous speech recognition of an inflected language using stems and endings. Speech Communication, Vol.49, No.6, pp.437-452

C. Chelba and F. Jelinek 1999. Structured language modeling for speech recognition In Proceedings of NLDB99, Klagenfurt, Austria

F. Richardson, M. Ostendorf, and J.R. Rohlicek. 1995. Lattice-based search strategies for large vocabulary speech recognition. Proc. ICASSP

M. Finke, J. Fritsch, D. Koll, A. Waibel. 1999. Modeling And Efficient Decoding Of Large Vocabulary Conversational Speech. In Proceedings of the EUROSPEECH99, Vol. 1, pp. 467–470, Budapest, Hungary, September 1999

Ircing P, Psutka J. 2002. Lattice Rescoring in Czech LVCSR System Using Linguistic Knowledge. International Workshop Speech and Computer SPECOM2002, St. Petersburgh, Russia. pp. 23-26.

Wikipedia 2007. http://en.wikipedia.org/wiki/Dravidian_languages

H. Jiang. 2005. Confidence measures for speech recognition: A survey, Speech Communication 45 (2005), pp. 455-470

Schaaf, T., Kemp, T. 1997. Confidence measures for spontaneous speech recognition. Proc. of International Conference on Acoustics, Speech and Signal Processing, pp. 875-878.

Wessel, F., Macherey, K., Ney., H. 1999. A comparison of word graph and N-best list based confidence measures. Proc. of European Conference on Speech Communication Technology, pp. 315-318

Rahim, M.G., Lee, C.-H. 1997. String-based minimum verification error (SB-MVE) training for speech recognition. Computer Speech Language 11, 147-160.

HTK website. 2007. http://htk.eng.cam.ac.uk/

Strassel, S., Kolar, J., Song Z., Barclay L., Glenn M. 2005. Structural Structural Metadata Annotation: Moving Beyond English. Proc. of European Conference on Speech Communication Technology, Lisbon.

Noise as a Tool for Spoken Language Identification

.

Sunita Maithani
Scientific Analysis Group,
Defense Research & Development Organization,
Metcalfe House, Delhi-110054, India.
E-mail: ysmaithani58@yahoo.com

J. S. Rawat
.Scientific Analysis Group,
Metcalfe House, Delhi-110054, India.
Defense Research & Development Organization,

Abstract

Segmental SNR (Signal to Noise Ratio) is considered to be a reasonable measure of perceptual quality of speech. However it only reflects the distortion in time dependent contour of the signal due to noise. Objective Measures such as Log Area Ratio (LAR), Itakura-Saitio Distortion (IS), Log-Likelihood Ratio (LLR) and Weighted Spectral Slope (WSS) are better measures of perceptual speech quality as they represent deviation in the spectrum. Noise affects the speech time contour and the corresponding frequency content. Different languages have some peculiar characteristics due to variation in the phonetic content and their distribution. Distortion introduced by noise and application of enhancement algorithm varies for different phonemes. In this paper a novel idea of using noise and speech enhancement as means of identifying a language is presented, using objective measures of speech quality. Study is done on three spoken Indian regional languages namely Kashmiri, Bangla and Manipuri, when corrupted by white noise. It is found that the objective measures of noisy speech, when determined using corresponding clear and enhanced speech are different for different languages over a range of SNR, giving clue to the type of the language in use.

1. Introduction

Speech is a signal which easily gets corrupted as it comes in contact with the environment. Except in sound-proof rooms used in studios, it is not possible to find such ideal noise free conditions in practice. Although a large number of noises exist in environment, broadly they can be classified into Factory, Babble, Engine, White and Channel noises etc. However most common kind of noise encountered is white noise, may it be in communication systems due to channel or generated in the equipment due to thermal or other electronic sources or combination of noises due to Central Limit Theorem (Aleksandr Lyapunov, 1901). Noise thus corrupts the speech, causing listener's fatigue and deteriorating performance of speech systems. Application of Speech enhancement or noise cancellation algorithms alleviates such problems to some extent. In literature several speech enhancement techniques exist. Though most traditional algorithms are based on optimizing mathematical criteria, they are not well correlated with speech perception and have not been as successful in preserving or improving quality in all regions of speech, especially transitional and unvoiced. Performance is also influenced by the specific type of noise, specific SNR, noise estimate updates and algorithm parameter settings. Spectral Subtraction technique of speech enhancement is popular and is still widely used as front end to speech systems for its simplistic nature and high quality performance except at very low SNRs (J. Lim, 1983).

Variety of languages exists in Indian region, with Dravidian, Tibeto-Burman, Indo-European, Indo-Aryan and Indo Iranian background. Mostly Indian languages are phonetic in nature that is there is one to one correspondence between sounds and the representative alphabet, and combining them creates similar kind of sounds. However different languages vary in its perceptibility due to

differences in its phonetic contents and variations in distribution of different phonemes, stress level distribution among phonemes and of course intonation pattern, nasality usage, allophonic variants, contextual, phonotactic, or coarticulatory constraints etc.

Introduction of noise in speech distorts the speech spectrum and affects its phonetic perceptibility differently, due to the factors mentioned above. Enhancement of noisy speech though reduces the noise and subsequent irritation, but generally results in distortion of the speech spectrum. The kind and amount of distortion in the spectrum of enhanced speech will depend on the particular enhancement technique applied, and the SNR of the noisy speech. Therefore different types of speech units will get affected differently by the noise and subsequent enhancement.

In this paper, a novel work on identification of spoken languages, based on effect of distortion introduced by white noise in the phonetic contents of different Indian Regional languages namely Kashmiri, Bangla and Manipuri is reported. This kind of approach is not found in the literature for any other language as well. Effect of Speech enhancement technique namely spectral subtraction on noisy speech of these languages is also studied at different levels of segmental SNR. White noise has been considered for noisy spoken language, as it affects all frequency components of speech uniformly. The distortion introduced in the resulting speech is measured by estimating objective measures of perceptual speech quality such as LLR, LAR, IS and WSS (Hansen and Pellom, 1998). The variation of these estimated objective measures of the spectral distortion, with regard to a particular language, is studied and analyzed, to see language specific effects of the noise and enhancement algorithm, in order to provide clue to the identity of language in use.

The paper has been organized in the following form: Section 2 gives details of Spectral Subtraction technique of enhancement used. Section 3 gives a comparative study of phonotactics of the three languages i.e. Kashmiri, Bangla and Manipuri in brief. Section 4 introduces the objective measures used, namely LAR, IS, LLR and WSS. Section 5 describes the Results and discussion. Section 6 gives conclusions.

2. Spectral Subtraction

This technique of speech enhancement is computationally very efficient, particularly for stationary noise or slowly varying non-stationary noise. Spectral subtraction is a noise suppression technique used to reduce the effects of added noise in speech. It estimates the power of clean speech by explicitly subtracting the estimated noise power from the noisy speech power. This of course assumes that the noise and speech are uncorrelated and additive in the time domain. Also, as spectral subtraction based techniques necessitate estimation of noise during regions of non-speech activity, it is supposed that noise characteristics change slowly. However, because noise is estimated during speech pauses, this makes the method computationally efficient. Unfortunately, for these reasons, spectral subtraction is beset by a number of problems. First, because noise is estimated during pauses the performance of a spectral subtraction system relies upon a robust noise/speech classification system. If a misclassification occurs this may result in a misestimating of the noise model and thus a degradation of the speech estimate. Spectral subtraction may also result in negative power spectrum values, which are then reset to non-negative values. This results in residual noise known as musical noise. In a speech enhancement application it has been shown that, at 5 dB SNR, the quality of the speech signal is improved without decreasing intelligibility. However, at lower SNR speech this performance reduces rapidly. When used in Automatic Speech Recognition (ASR), the trade-off between SNR improvement and spectral distortion is important. To provide a mathematical description of the spectral subtraction technique, we write the spectrum of the noisy speech y (t) in terms of that of the clean speech x (t) and additive noise n (t) (the simplest acoustic distortion model):

$$y(t) = x(t) + n(t) \quad - (1)$$

The enhancement is explained in the following formula (Berouti et al., 1979).

$$\hat{X}(w) = \left[|Y(w)|^{\lambda} - \alpha \left| \hat{N}(w) \right|^{\gamma} \right]^{1/\gamma} e^{j\theta_y(w)} \quad - (2)$$

$\hat{X}(w)$ and $Y(w)$ are DFT (discrete fourier transform) of the enhanced and noisy signal. N (w) is estimate of noise and θ_Y phase of original signal. λ is 2 for working in power spectrum domain and α is the over subtraction factor.

3. Characteristics of Manipuri, Bangla and Kashmiri spoken languages

Different Indian regional languages have certain linguistic background of their own and later have added certain foreign loan words. Their phonotactics and grammar is also quite distinct Following are features of above spoken languages:

Manipuri: It is a Tibeto-Burman language. Tone is used to convey phonemic distinction. Aspirates are present. High frequency of the velar nasal is particularly striking. Grammatical gender is missing. The normal order of words in a sentence is SOV-subject, object, verb, though this is not always and everywhere rigorously observed. Tibeto-Burman words are monosyllables. Phonological system of Manipuri can be categorized into two groups – segmental phonemes and supra-segmental phonemes. Segmental phoneme includes vowels and consonants and supra-segmental phoneme includes tone and juncture. All the six Manipuri vowels can occur in initial, medial and final position. There are six diphthong like sounds in Manipuri. They are

- (/əy/,/ay/, /əw/ ,/oy/, /uy/, /aw/)

There are 24 consonant phonemes in Manipuri p,t,k, ph,th,kh,m, n,ŋ,c,s,l, h,w,y,b d,g,bh, dh,gh,j, jh,r . Among these the last 9 voiced sounds are borrowed from other languages and they cannot occur in the initial and final position. Only four phonemes can occur in the second element of the cluster. They are w, y, r and l. It can occur only in the initial and medial position of a word. There are two types of tone in the language level and falling tone. Juncture, other than phonetic features, has a phonemic status.

Bangla: An Indo-Aryan language. Standard colloquial Bengali contains 35 essential phonemes. 5 non-essential phonemes which occur only as variants of other sounds or in borrowed foreign words & not used by all speakers. The ten aspirated stops and affricates are characteristics and essential sounds of the language. They are not simple but compounds.

Seven vowel phonemes occur with their opposite nasal phoneme. All may be long or short. Length is not considered to be phonemic. There is one 1st person pronoun, three 2nd person pronouns and three pairs of 3rd person pronouns with polite, informal, singular, plural discrimination. Pronoun and verb have no gender discriminatory word. Most of the sentences don't explicitly use verbs. Verbs are inflected in person (1st, 2nd, 3rd), in degrees of politeness (intimate, familiar, respectful), and in tense (past, present, future). Plural can be inflected by adding suffix –ra, -der, -era, -diger, -guli, -gulo, -gana. The dominant word order in Modern Bengali sentences is:

Subject + Indirect object + Direct object + Oblique object + Verb.

Kashmiri: All the vowels have a nasal counterpart. Nasalization is phonemic in Kashmiri. Palatalization is phonemic in Kashmiri. All the non-palatal consonants in Kashmiri can be palatalized. There are eight pairs of short and long vowels. Kashmiri is a syllable-timed language, sometimes; individual words are stressed for emphasis. There are four major types of intonational patterns: (1) High - fall, (2) High - rise, (3) Rise &fall, (4) Mid - level. Intonations have syntactic rather than emotional content. Vowels /ə/, /o/, /ɔ:/ do not occur in the word final position. The short vowels /ɨ/, /e/, /u/, and / ɔ/ do not occur in the word-initial position. Usually the semi-vowel /y/ is added in the initial position of the words beginning with /i/, /i:/, /e/ and /e:/. Similarly, the semi-vowel /v/ is added to the words beginning with /u/, and /u:/. Vowel sequences usually do not occur in Kashmiri. Word initial consonant clusters are not as frequent as the word medial consonant clusters. Kashmiri has (C)(C)V(C)(C) syllable structure.

4. Objective methods of speech quality measure

In general speech enhancement or noise reduction is measured in terms of improvement in SNR, but in reality, this may not be the most appropriate performance criteria for improvement of perceptual speech quality. Humans do have an intuitive understanding of spoken language quality, however this may not be easy to quantify. In a number of studies, it has been shown that impact of noise on degradation of speech quality is non uniform. An objective speech quality measure shows, the level of distortion for each frame, across time. Since speech frequency content varies, across time, due to sequence of phonemes, needed to produce the sentence, impact of background distortion will also vary, causing some phone classes to get more effected than others, when produced in a noisy environment. Objective methods rely on mathematically based measure between reference signal and the signal under consideration. The objective measures are based on different parametric representation of the speech, and differ due to inclusion or non-inclusion of various parameters and the different weightage given to them, in order to imitate auditory model and perception as closely as possible. The details of each one is given below.

Itakura-Saitio Distortion Measure (IS): If for an original clean frame of speech, linear prediction (LP) coefficient vector is $\vec{a}_{\Phi}$, correlation matrix is R_{Φ}. And for processed speech LP coefficient vector is $\vec{a}_d$, correlation matrix is R_d, then Itakura-Satio distortion measure is given by,

$$d_{IS}\left(\vec{a}_d\ ,\vec{a}_\phi\right)=\left[\frac{\sigma_\phi^2}{\sigma_d^2}\right]\left[\frac{\vec{a}_d R_\phi \vec{a}_d^T}{\vec{a}_\phi R_\phi \vec{a}_\phi^T}\right]+\log\left[\frac{\sigma_d^2}{\sigma_\phi^2}\right]-1 \qquad \text{- (3)}$$

Where σ_d^2 and σ_Φ^2 represents the all-pole gains for the processed and clean speech frame respectively.

Log-Likelihood Ratio Measure (LLR): The LLR measure is also referred to as the Itakura distance. The LLR measure is found as follows,

$$d_{LLR}\left(\vec{a}_d\ ,\vec{a}_\phi\right)=\log\left[\frac{\vec{a}_d R_\phi \vec{a}_d^T}{\vec{a}_\phi R_\phi \vec{a}_\phi^T}\right] \qquad \text{- (4)}$$

Log-Area-Ratio Measure (LAR): The LAR measure is also based on dissimilarity of LP coefficients between original and processed speech signals. The log-area-ratio parameters are obtained from the pth order LP reflection coefficients for the original $r_\Phi(j)$ and processed $r_d(j)$ signals for frame j. The objective measure is formed as follows,

$$d_{LAR}=\left|\frac{1}{M}\sum_{i=1}^{M}\left[\log\frac{1+r_\phi(j)}{1-r_\phi(j)}-\log\frac{1+\hat{r}_d(j)}{1-\hat{r}_d(j)}\right]\right|^{\frac{1}{2}} \qquad \text{- (5)}$$

Weighted Spectral Slope Measure (WSS): The WSS measure by Klatt (1982) is based on an auditory model, in which 36 overlapping filters of progressively larger bandwidth are used, to estimate the smoothed short-time speech spectrum. The measure finds a weighted difference between the spectral slopes in each band. The magnitude of each weight reflects whether the band is near a spectral peak or valley, and whether the peak is the largest in the spectrum. A per-frame measure in decibel is found as

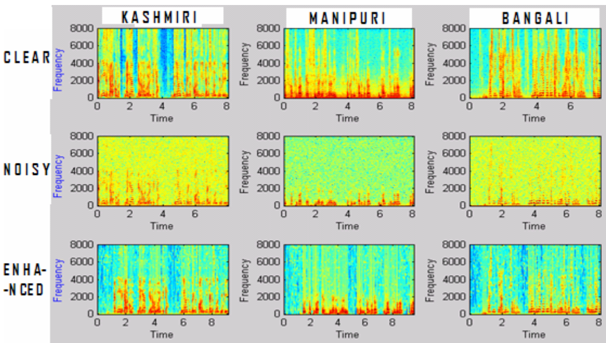
$$d_{WSS}(j)=K_{spl}\left(K-\hat{K}\right)+\sum_{k=1}^{36}w_a(k)\left(S(k)-\hat{S}(k)\right)^2 \qquad \text{- (6)}$$

where K, $\hat{K}$ are related to overall sound pressure level of the original and enhanced utterances, and K_{spl} is a parameter which can be varied to increase overall performance.

5. Results and Discussion

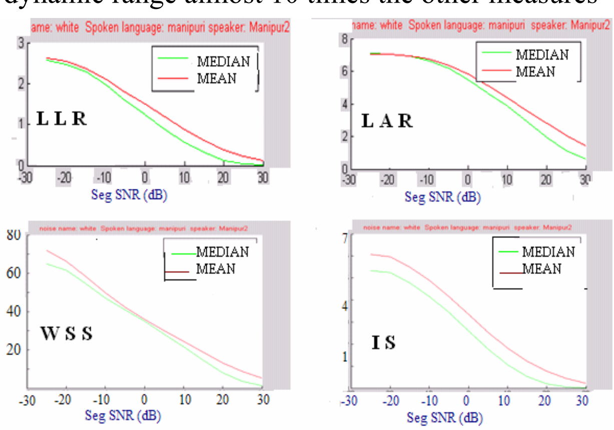
Sentences Spoken by 30 native speakers for each language namely Manipuri, Bangla and Kashmiri were recorded at 16 KHz. Noisy speech with white noise was simulated with segmental SNR from 30 dB to -20 dB. Objective measures i.e. IS, LAR, LLR and WSS are computed for each frame, with length ~ 512 samples. In first experiment these measures are computed for the noisy speech with reference to the corresponding clean speech sentence, whereas in second experiment the objective measures are computed using enhanced speech and the corresponding noisy speech for different sentences of the languages. Estimates of these measures are determined for the complete sentence using two methods, namely 5% trim

mean and median of their values computed for each frame. Spectral subtraction method of enhancement is applied to obtain enhanced speech from the noisy speech sentences. For 10 dB SegSNR noisy speech, the spectrograms of the speech in three languages corresponding to Clean, Noisy and Enhanced, is shown in figure 1. It is observed through the spectrograms, that the noise has affected the three languages differently.



“Figure 1. Speech Spectrograms descriptions Rows: 1st-Clear, 2nd-Noisy, 3rd-Enhanced; Columns: 1st-Kashmiri, 2nd -Manipuri, 3rd -Bangla”

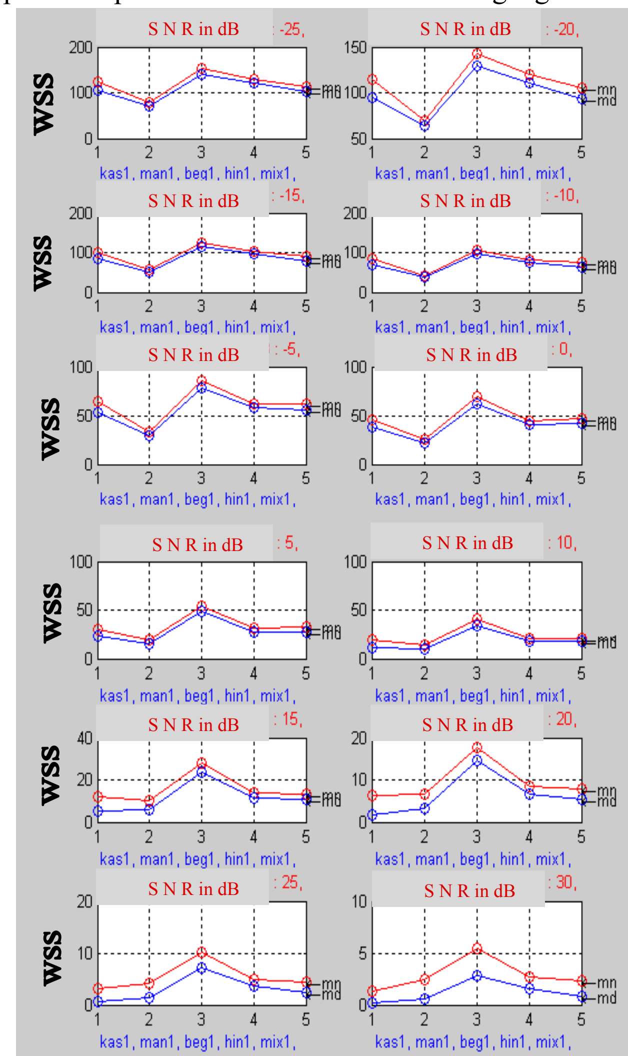
Estimates of LLR, LAR, IS and WSS are computed for SNR range 30 dB to -30 dB for different speech sentences in the three languages using noisy and clear and then enhanced and noisy speech. It is seen that WSS measure has the widest dynamic range almost 10 times the other measures



“Figure 2. LLR, LAR, IS and WSS estimates vs. SNR plots in Manipuri Speech with experiment-1.”

as shown in figure 2. of experiment -1, using Manipuri Speech. Thus it can be seen, that WSS is most suitable for the studies of distortion effects, of noise and enhancement algorithm, on different spoken languages.

WSS estimates of noisy speech, at different SNR are computed, as in experiment-1, and plotted in figure 3. It is observed that Manipuri is having lowest WSS estimate followed by Kashmiri and then highest for Bangla. This trend is more prominent particularly for low SNRs. The other points of plot are for Hindi and mixed languages.



“Figure 3. Plots of WSS estimates (y axis) as in experiment 1. , for different SNRs in dB i.e. -25, -20, -15, -15, -10, 0, 5, 10, 15, 20, 25, 30 for Kashmiri, Manipuri, Bangla, Hindi and mixed languages (denoted in x axis by 1, 2, 3, 4, 5 respectively)”

In experiment 2, WSS and LAR estimates are computed for enhanced speech, with reference to the corresponding noisy speech, for the three languages namely Kashmiri, Manipuri, and Bangla. The enhanced speech is obtained after application of

spectral subtraction algorithm on noisy speech of different SNRs, ranging from 30 dB to -20 dB in steps of 5 dB. The mean and median estimates of the WSS for the 2nd experiment are shown in table 1. Here also the WSS estimate is lowest for Manipuri, followed by Kashmiri and Bangla is the highest. This trend is more prominent for low SNRs.

SNR in dB	Language	WSS Estimates	
		Median	Mean
30	Kashmiri	36.22793	42.52874
	Manipuri	32.12041	36.83813
	Bangla	38.06589	42.30879
25	Kashmiri	40.25494	45.34821
	Manipuri	34.70880	39.67888
	Bangla	42.705033	48.92245
20	Kashmiri	46.72147	53.09616
	Manipuri	38.03188	42.95194
	Bangla	51.42718	57.53441
15	Kashmiri	53.70700	60.94677
	Manipuri	45.73857	51.09685
	Bangla	60.85805	67.17440
10	Kashmiri	65.43084	71.24645
	Manipuri	58.61265	71.94426
	Bangla	70.73388	77.70258
0	Kashmiri	87.72349	92.32025
	Manipuri	71.26169	78.03224
	Bangla	92.23746	97.70964
-5	Kashmiri	94.50976	97.43755
	Manipuri	78.38978	83.14540
	Bangla	101.4064	104.6625
-10	Kashmiri	98.70403	100.8097
	Manipuri	85.42304	91.05538
	Bangla	105.2050	109.2263
-20	Kashmiri	101.8426	106.9107
	Manipuri	96.24993	101.5472
	Bangla	109.1610	112.9643

"Table 1. Median and Mean estimates of WSS for Enhanced speech in Kashmiri, Manipuri and Bangla for SNRs -30 dB to 20 dB as in Experiment 2."

6. Conclusion

In this paper a study is done for possibility of using LLR, LAR, IS and WSS as objective measures of speech quality, for discrimination of Indian regional languages namely Kashmiri, Manipuri and Bangla. This is done by computing estimates of of these objective measures for noisy speech with white noise for the above spoken languages and at SNRs -30 dB to 30 dB. First these measures are computed for noisy speech with reference to corresponding clear speech and then for the enhanced speech with reference to the corresponding noisy speech. WSS has proved to be the most useful measure used due to its wider dynamic range. The two estimates of WSS do provide clue to the type of language in use due to differences in its phonetic content. The discrimination provided is highest at lower SNRs. The estimate being lowest for Manipuri, and highest for Bangla. The reason could be attributed to the presence of weaker speech units in relatively higher concentration, in the language with higher WSS estimates compared to others; as the speech parameters under consideration for them, would undergo higher distortion under the influence of noise.

7. Acknowledgement

The authors are thankful to Director, Dr. P K Saxena and Dr S S Bedi for encouraging us to carry out this work and allowing presentation of the paper.

References

A.F Martin F.J. Godman and R.E.Wohlford. 1989. *Improved automatic language identification in noisy speech.* Proc Int Conf Acoust. Speech, and Signal Processing . May. 528-531

John H.L. Hansen and Bryan L. Pellom. Nov 1998. *Speech Enhancement and Quality Assessment:* A Survey, IEEE Signal Processing magazine

J. Lim. 1983. *Speech Enhancement.* Prentice Hall Englewood Cliffs, NJ.

Klatt, D.1982. *Prediction of perceived phonetic distance from critical-band spectra.* Proc. of IEEE Int. Conf on ASSP, 1278-1281.

M. Berouti, R. Schwartz and J. Mahoul.1979. *Enhancement of Speech corrupted by acoustic noise.* ICASSP, 208-211

Identifying Real or Fake Articles: Towards better Language Modeling

Sameer Badaskar
School of Computer Science
Carnegie Mellon University
Pittsburgh PA, United States
sbadaska@cs.cmu.edu

Sachin Agarwal
School of Computer Science
Carnegie Mellon University
Pittsburgh PA, United States
sachina@cs.cmu.edu

Shilpa Arora
School of Computer Science
Carnegie Mellon University
Pittsburgh PA, United States
shilpaa@cs.cmu.edu

Abstract

The problem of identifying good features for improving conventional language models like trigrams is presented as a classification task in this paper. The idea is to use various syntactic and semantic features extracted from a language for classifying between real-world articles and articles generated by sampling a trigram language model. In doing so, a good accuracy obtained on the classification task implies that the extracted features capture those aspects of the language that a trigram model may not. Such features can be used to improve the existing trigram language models. We describe the results of our experiments on the classification task performed on a Broadcast News Corpus and discuss their effects on language modeling in general.

1 Introduction

Statistical Language Modeling techniques attempt to model language as a probability distribution of its components like words, phrases and topics. Language models find applications in classification tasks like Speech Recognition, Handwriting Recognition and Text Categorization among others. Conventional language models based on n-grams approximate the probability distribution of a language by computing probabilities of words conditioned on previous n words as follows

$$P(s) \approx \prod_{i=1}^{m} p(w_i | w_{i-n+1}, \ldots, w_{i-1}) \quad (1)$$

In most applications, lower order n-grams (such as bigram or trigram) are used but they are an unrealistic approximation of the underlying language. Higher order n-grams are desirable but they present problems concerning data sparsity. On the other hand, low order n-grams are incapable of representing other aspects of the language like the underlying topics, topical redundancy etc. In order to build a better language model, additional features have to be augmented to the existing language model (e.g. a trigram model) which capture those aspects of the language that the trigram model does not. Now, one way to test the goodness of a feature under consideration is to use it in a framework like an exponential model (Rosenfeld, 1997; Cai et al., 2000) and note the improvement in perplexity. An alternative way (Eneva et al., 2001) is as follows: Let L be the language and $\tilde{L}$ be an approximation of the language obtained by sampling the trigram language model. Also, let X be a piece of text obtained from either L or $\tilde{L}$. Let $y = h(f(X))$ such that $y = 1$ if $X \in L$ and $y = 0$ if $X \in \tilde{L}$ where $f(.)$ is the computed feature and $h(.)$ is the hypothesis function (a classifier like AdaBoost, SVM etc). If $Pr[y = h(f(x))]$ is found to be sufficiently high, it means that the feature $f(x)$ is able to distinguish effectively between the actual language L and the approximate language $\tilde{L}$. In other words, $f(x)$ captures those features of the language that are complementary to the ones captured by the trigram model and therefore $f(x)$ is a *good* feature to augment the trigram language model with.

The formalism explained previously can be interpreted as a classification task in-order to distinguish

between *Real* articles and *Fake* articles. Articles of different lengths drawn at random from the Broadcast News Corpus (BNC)[1] are termed as *Real* articles (from language L). Articles generated by sampling the trigram model trained on the same corpus are termed as *Fake* articles (language $\tilde{L}$). These articles together form the training data for the classifier to associate the features with the classification labels (*real* or *fake*) where the features are computed from the text. The features that give high classification accuracy on the test set of articles are considered good candidates for adding to the trigram model. Furthermore, the confidence that the classifier attaches to a classification decision can be used to compute the perplexity.

In this paper, a classification-task based formalism is used to investigate the goodness of some new features for language modeling. At the same time features proposed in the previous literature on language modeling are also revisited (Cai et al., 2000) Section 2 discusses various syntactic and semantic features used for the classification task, Section 3 gives details about the experiments conducted and the classification results obtained and finally, Section 4 concludes the paper by discussing the implications of the classification results on language modeling with pointers to improvements and future work.

2 Feature Engineering

To differentiate a *real* article from a *fake* one, the empirical, syntactic and semantic characteristics of a given article are used to compute the features for the classification task. The various types of features that were experimented are as follows:

2.1 Empirical Features

Empirical features are based on the statistical analysis of both the *real* and *fake* articles. They include the count of uncommon pairs of words within an article, the ratio of perplexity of trigram and quadgram models for a given article and the nature of the POS tags that occur at the start and end of sentences in an article.

[1]http://www.cs.cmu.edu/ roni/11761-s07/project/LM-train-100MW.txt.gz

Ratio of Perplexity of trigram and quad-gram models

Given an article, the ratio of its perplexity for a trigram model to a quad-gram model is computed. The trigram and quad-gram models are both trained on the same BNC corpus. Both *real* and *fake* articles would give a low perplexity score for the tri-gram model but for the quad-gram model, *real* articles would have significantly lower perplexity than the *fake* articles. This implies that the ratio of trigram to quad-gram perplexities would be lower for a *fake* article than for a *real* article. In other words, this ratio is similar to computing the likelihood ratio of an article w.r.t the trigram and quad-gram models. The histogram in Figure 1 shows a good separation in the distribution of values of this feature for the *real* and *fake* articles which indicates the effectiveness of this feature. A quadgram language model is a better approximation of *real* text than a trigram model and by using this as a feature, we are able to demonstrate the usefulness of the classification task as a method for identifying good features for language modeling. In the subsequent sections, we investigate other features using this classification framework.

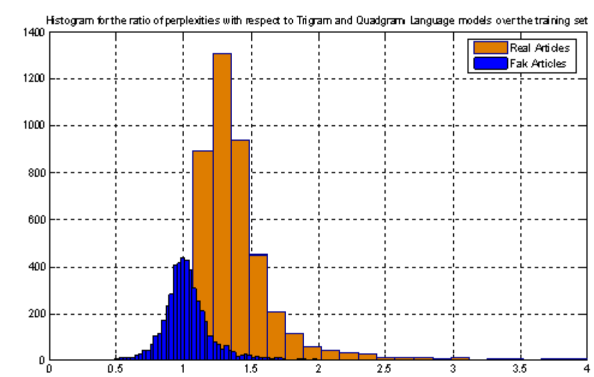


Figure 1: Histogram for the ratio of perplexities with respect to Trigram and Quadgram Language models over the training set

Count of uncommon pairs of words

Content words are the frequently occurring words in the corpus excluding the stop-words. All the words in corpus are ranked according to frequency of their occurrence and content words are defined to be the words with rank between 150 and 6500. A list of common content word pairs (pairs of content words

atleast 5 words apart) is prepared from the *real* corpus by sorting the list of content word pairs by their frequency of occurrence and retaining those above a certain threshold. For a given article, a list of content word pairs is compared against this list and word pairs not in this list form the set of uncommon word pairs.

A *real* article is expected to have lesser number of uncommon content-word pairs than *fake* articles. When normalized by the total number of word pairs, we get the probability of finding an uncommon content-word pair in an article. This probability is greater for *fake* articles than the *real* articles and we use this probability as a feature for the classification task.

Start and End POS Tags

Certain POS tags are more probable than others to appear at the beginning or end of a *real* sentence. This characteristic of *real* text could be used as a feature to distinguish *real* articles from *fake*. The distribution of POS tags of the first and last words of the sentences in an article is used as a feature. Our experiments show that this feature had very little effect in the overall contribution to the classification accuracy over the development set.

2.2 Syntactic Features

These features are derived from the parse structure of the sentence. It is hypothesized that *real* sentences tend to be grammatical while the same may not be the case for *fake* sentences. An objective measure of the grammaticality of a sentence can be obtained by running it through a statistical parser. The log-likelihood score returned by the parser can be used to judge the grammaticality of a sentence and thus determine whether it is *fake* or *real*. The Charniak Parser (Charniak, 2001; Charniak, 2005) was used for assessing the grammaticality of the articles under test. Given an article containing sentences $S_1, S_2, \ldots, S_N$ with lengths $L_1, L_2, \ldots, L_N$, we compute the parser log-likelihood scores $P(S_1), P(S_2), \ldots, P(S_N)$. The overall grammaticality score for an article is given by

$$P_{Gram} = \frac{\sum_{i=1}^{N} L_i P(S_i)}{\sum_{i=1}^{N} L_i} \qquad (2)$$

The grammaticality score was normalized using the average and standard deviation over the entire training set. This feature gave small improvement in terms of classification accuracy. There may be several reasons for this: (1) Our training data consisted of spoken transcripts from a broadcast news corpus whereas the Charniak Parser was trained on a different domain (Wall Street Journal) and (2) The parser was trained on mixed case text where as the data we used was all upper case.

2.3 Semantic Features

Real articles contain sentences with correlated pairs of content-words and sentences that are correlated with each other. An article with such sentence/word correlations is said to be semantically coherent. Owing to the use of only the short term word history for computing the probability distribution of a language, a trigram model fails to model semantic coherence and we exploit this fact for the classification task. Specifically, we intend to model both intra-sentence and inter-sentence semantic coherence and use them as features for classification.

Intra-sentence Coherence

To model the intra-sentence word correlations, we use Yule's Q-statistic (Eneva et al., 2001). The word correlations are learned from the BNC corpus as well as the *fake* corpus. The coherence score for an article is defined as the sum of the correlations between pairs of content words present in the article. The coherence score for an article is normalized by the total number of content-word pairs found in the article. Since the trigram and quad-gram language model can capture short distance coherences well, coherences between distant words can be used to differentiate between *real* and *fake* articles. The Yule Q-statistic is calculated for every pair of content words, which are atleast 5 words apart within a sentence, both in the *real* and *fake* corpus.

The articles are scored according to content word-pair correlations learned from the *real* as well as *fake* corpus. Each article is given two scores, one for the word-pair correlations from *real* articles and other for the word-pair correlations from *fake* articles. For a *real* article, the *real* word-pair correlation score would be relatively higher compared to the *fake* word-pair correlation score (and vice-versa

for a *fake* article).

Modeling Topical Redundancy (Inter-sentence Coherence)

A characteristic of *real* articles is that they tend to be cohesive in terms of the topic under discussion. For example, a news-article about a particular event (topic) would have several direct or indirect references to the event. We interpret this as some sort of a *redundancy* in terms of the information content which we term as Topical Redundancy. The *fake* articles would not exhibit such a redundancy. If a *real* article is transformed to another *representation space* where some form of *truncation* is applied, on transformation back to the original space, the amount of information-loss may not be significant due to information redundancy. However, if the same process is applied on a *fake* article, the information-loss would be significant when transformed back to the original space. We intend to exploit this fact for our classification task.

Let $D_{W \times N}$ be an article represented in the form of a matrix, where W is the article vocabulary and N is the number of sentences in that article. Every term of this matrix represents the frequency of occurrence of a vocabulary word in a particular sentence. We construct a sentence-sentence matrix as follows:

$$A = D^T D \tag{3}$$

We now *transform* A into the Eigen-space using Singular Value Decomposition (SVD) which gives

$$A = USU^T \tag{4}$$

Here, $U_{N \times N}$ is the eigen-vector matrix and $S_{N \times N}$ is the diagonal eigen-value matrix. If we retain only the top K eigen-values from S , we get the truncated (lossy) form $S'_{K \times K}$. Thus the *truncated* form of A i.e. A' is

$$A' = US'U^T \tag{5}$$

We believe that the information loss $\parallel A - A' \parallel^2$ will not be significant in the case of *real* articles since the topical redundancy is captured in a very compact manner by the eigen-representation. However, in the case of a *fake* article, the loss is considerable. For a *real* article, the matrix would be less sparse than a *fake* article and so is the case for the reconstructed matrix. Therefore, the statistics - mean, median, minimum and maximum computed from the reconstructed matrix have higher values for *real* articles than a *fake* articles. We use these statistics as features for classifying the article. Figure 2 show the histograms of the statistics computed from the reconstructed matrix for the training set. As can be seen, there is a good separation between the two classes *fake* and *real* in all the cases. Using these features increased the classification accuracy by a significant amount as shown later. From another perspective, these features model the inter-sentence semantic coherence (Deerwester et al., 1990) within an article and this is consistent with our notion of topical redundancy as explained previously. The matrix package developed by NIST (Hicklin et al., 2005) was used for SVD.

3 Experimental Results

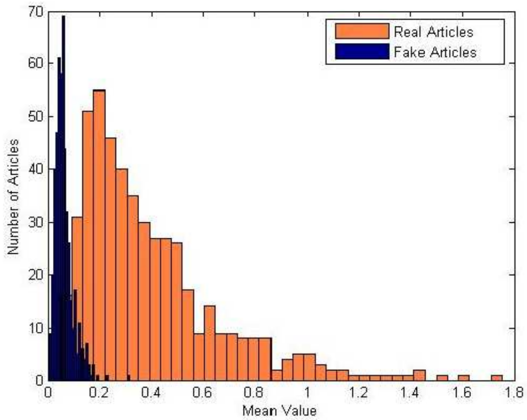
3.1 Data Distribution

The training data consisted of 1000 articles (500 *real* and 500 *fake*) obtained from Broadcast News Corpus (BNC) and the test set consisted of 200 articles (100 *real* and 100 *fake*). Additionally, a development dataset consisting of 200 articles and having the same distribution as that of the test dataset was used for tuning the parameters of the classifiers. To ensure that the training and test data come from the same article length distribution, the training data was resampled to have the same percentage of articles of a given length as in the test set. The article length distribution for both the training(resampled) and test datasets is shown in Tables 1 and 2.

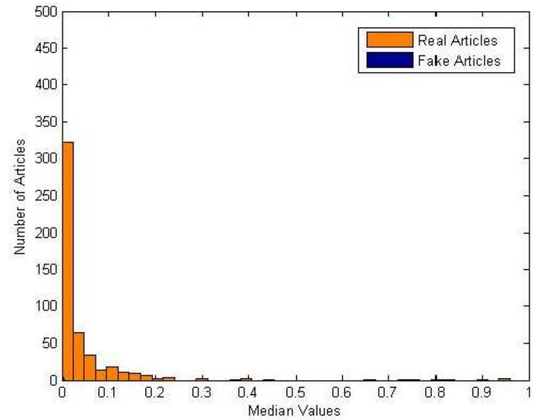
3.2 Classifier

Classifiers like AdaBoost (Freund et al., 1999) and Max-Entropy (Rosenfeld, 1997) models were used for the classification task.

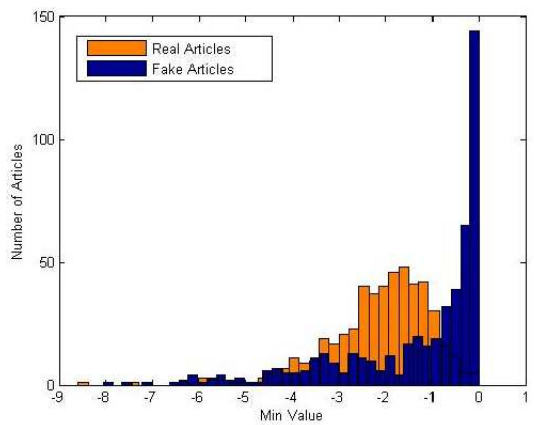
The number of iterations for AdaBoost was estimated using 5-fold cross-validation. Given a subset of features, Maxent classified 74.5% of the documents correctly compared to 82% for AdaBoost. Therefore, Adaboost was chosen as the classifier for further experiments.



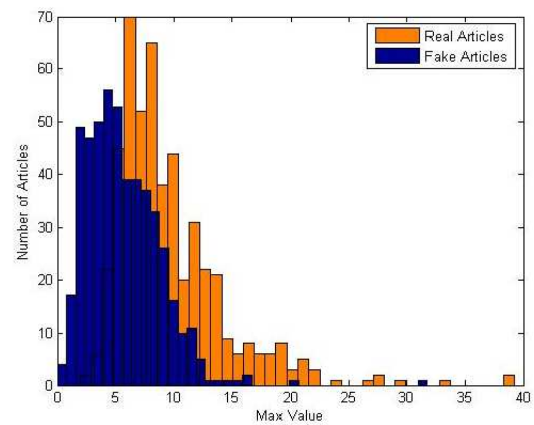
(a) Mean



(b) Median



(c) Minimum



(d) Maximum

Figure 2: Histograms of topical redundancy features computed over the training set. In (b) , the median values for the fake articles are close to zero and hence cannot be seen clearly.

3.3 Results and Discussion

We used two performance measures to evaluate our model. First is the accuracy which measures the number of articles correctly classified as *real* or *fake* and the second measure is the log-probability that the model assigns to the classification decision i.e. it measures the confidence the model has in its classification. Table 3 shows our experimental results on the syntactic, semantic and empirical features.

The combination of syntactic, semantic and empirical features gave an accuracy of 91.5% with an average log-likelihood of -0.22 on development data set. The accuracy on the test dataset was 87% with an average log-likelihood of -0.328.

4 Conclusions and Future Work

In this work, we have used a classification-task based formalism for evaluating various syntactic, semantic and empirical features with the objective of improving conventional language models. Features that perform well in the task of classifying *real* and trigram-generated *fake* articles are useful for augmenting the trigram model. Semantic features, such as topical redundancy, model long-range dependencies which are not captured by a trigram language model. Therefore, the semantic features contribute significantly to the classification task accuracy. Additionally, linguistic resources such as WordNet (WordNet, 1998) can be used to model

# Sentences per article	# Real Art.	# Fake Art.	% Total (Real & Fake)
1	938	940	19.76
2	440	471	9.58
3	502	474	10.26
4	507	533	10.94
5	497	525	10.75
7	431	524	10.05
10	475	479	10.04
15	482	421	9.50
20	421	446	9.12

Table 1: Distribution of article lengths for training dataset.

# Sentences per article	# Real Art.	# Fake Art.	% Total (Real & Fake)
1	20	20	20
2	10	10	10
3	10	10	10
4	10	10	10
5	10	10	10
7	10	10	10
10	10	10	10
15	10	10	10
20	10	10	10

Table 2: Distribution of article lengths for test dataset.

topical redundancy using synonyms and other inter-word dependencies. The semantic features we explored assume a single underlying topic for an article which may not be always true. An article can be a representation of different topics and we aim to explore this direction in future.

Feature Combination	Classification Accuracy	Avg. Log Likelihood
Syntactic	60.5%	-0.663
Semantic	79.5%	-0.510
Empirical	83.0%	-0.446
Semantic + Syntactic	80.0%	-0.553
Semantic + Empirical	86.0%	-0.410
Semantic + Syntactic + Empirical	91.5%	-0.220

Table 3: Performance of different features on the development-set.

References

Can Cai, Larry Wasserman and Roni Rosenfeld. 2000. *Exponential language models, logistic regression, and semantic coherence*. Proceedings of the NIST/DARPA Speech Transcription Workshop.

Eugene Charniak. 2001. *Immediate-Head Parsing for Language Models*. Proceedings of 39th Annual Meeting of the ACL, 124-131.

Eugene Charniak. 2005. ftp://ftp.cs.brown.edu/pub/ nlparser/parser05Aug16.tar.gz

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, New York.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman. 1990. *Indexing by Latent Semantic Analysis*. Journal of Japanese Society for Artificial Intelligence, 41(6).

Elena Eneva, Rose Hoberman and Lucian Lita. 2001. *Learning within-sentence semantic coherence*. Proceedings of the EMNLP 2001.

Yoav Freund and Robert E. Schapire. 1999. *A short introduction to boosting* Journal of Japanese Society for Artificial Intelligence, 14(5):771-780.

Joe Hicklin, Cleve Moler, Peter Webb. 2005. http://math.nist.gov/javanumerics/jama/

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Roni Rosenfeld. 1997. *A whole sentence maximum entropy language model*. In Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding, 1997.

WordNet: An Electronic Lexical Database, ISBN-13: 978-0-262-06197-1.

Multi-label Text Categorization with Model Combination based on F_1-score Maximization

Akinori Fujino, Hideki Isozaki, and **Jun Suzuki**
NTT Communication Science Laboratories
NTT Corporation
2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan 619-0237
{a.fujino,isozaki,jun}@cslab.kecl.ntt.co.jp

Abstract

Text categorization is a fundamental task in natural language processing, and is generally defined as a multi-label categorization problem, where each text document is assigned to one or more categories. We focus on providing good statistical classifiers with a generalization ability for multi-label categorization and present a classifier design method based on model combination and F_1-score maximization. In our formulation, we first design multiple models for binary classification per category. Then, we combine these models to maximize the F_1-score of a training dataset. Our experimental results confirmed that our proposed method was useful especially for datasets where there were many combinations of category labels.

1 Introduction

Text categorization is a fundamental task in such aspects of natural language processing as information retrieval, information extraction, and text mining. Since a text document often belongs to multiple categories in real tasks such as web pages and international patent categorization, text categorization is generally defined as assigning one or more predefined category labels to each data sample. Therefore, developing better classifiers with a generalization ability for such multi-label categorization tasks is an important issue in the field of machine learning.

A major and conventional machine learning approach to multi-label categorization is based on binary classification. With this approach, we assume the independence of categories and design a binary classifier for each category that determines whether or not to assign a category label to data samples. Statistical classifiers such as the logistic regression model (LRM), the support vector machine (SVM), and naive Bayes are employed as binary classifiers (Joachims, 1998).

In text categorization, the F_1-score is often used to evaluate classifier performance. Recently, methods for training binary classifiers to maximize the F_1-score have been proposed for SVM (Joachims, 2005) and LRM (Jansche, 2005). It was confirmed experimentally that these training methods were more effective for obtaining binary classifiers with better F_1-score performance than the minimum error rate and maximum likelihood used for training conventional classifiers, especially when there was a large imbalance between positive and negative samples. In multi-label categorization, macro- and micro-averaged F_1-scores are often used to evaluate classification performance. Therefore, we can expect to improve multi-label classification performance by using binary classifiers trained to maximize the F_1-score.

On the other hand, classification frameworks based on classifier combination have also been studied in many previous works such as (Wolpert, 1992; Larkey and Croft, 1996; Ting and Witten, 1999; Ghahramani and Kim, 2003; Bell et al., 2005; Fumera and Roli, 2005), to provide better classifier systems. In the classifier combination research field, it is known that weighted linear combinations of multiple classifiers often provide better classification performance than individual classifiers.

We present a classifier design method based on the combination of multiple binary classifiers to improve multi-label classification performance. In our framework, we first train multiple binary classifiers for each category. Then, we combine these binary classifiers with weights estimated to maximize micro- or macro-averaged F_1-scores, which are often used for evaluating multi-label classifiers. To estimate combination weights, we extend the F_1-score maximization training algorithm for LRM described in (Jansche, 2005). Using three real text datasets, we show experimentally that our classifier design method is more effective than the conventional binary classification approaches to multi-label categorization.

Our method is based on a binary classification approach. However, Kazawa et al. (2005) proposed a method for modeling a map directly from data samples to the combination of assigned category labels, and confirmed experimentally that the method outperformed conventional binary classification approaches. Therefore, we also compare our method with the direct mapping method experimentally.

2 F_1-score Maximization Training of LRM

We first review the F_1-score maximization training method for linear models using a logistic function described in (Jansche, 2005). The method was proposed in binary classification settings, where classifiers determine a class label assignment $y \in \{1, 0\}$ for a data sample represented by a feature vector $\boldsymbol{x}$. Here, $y^{(n)} = 1\ (= 0)$ indicates that the class label is assigned (unassigned) to the nth feature vector $\boldsymbol{x}^{(n)}$.

The discriminative function of a binary classifier based on a linear model is often defined as

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}_1^t \boldsymbol{x} + \theta_0, \quad (1)$$

where $\boldsymbol{\theta} = (\theta_0, \boldsymbol{\theta}_1^t)^t$ is a model parameter vector, and $\boldsymbol{\theta}_1^t \boldsymbol{x}$ implies the inner product of $\boldsymbol{\theta}_1$ and $\boldsymbol{x}$. A binary classifier using $f(\boldsymbol{x}; \boldsymbol{\theta})$ outputs a predicted class label assignment $\hat{y}$ for $\boldsymbol{x}$ as $\hat{y}^{(n)} = 1\ (= 0)$ when $f(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}) \geq 0\ (< 0)$.

An LRM is a binary classifier that uses the discriminative function $f(\boldsymbol{x}; \boldsymbol{\theta})$. In this model, the class posterior probability distribution is defined by using a logistic function:

$$g(z) = \{1 + \exp(-z)\}^{-1}. \quad (2)$$

That is, $P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = g(f(\boldsymbol{x}; \boldsymbol{\theta}))$ and $P(y = 0|\boldsymbol{x}; \boldsymbol{\theta}) = 1 - P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = g(-f(\boldsymbol{x}; \boldsymbol{\theta}))$. The LRM determines that $y^{(n)} = 1\ (= 0)$ when $P(y = 1|\boldsymbol{x}^{(n)}; \boldsymbol{\theta}) \geq 0.5\ (< 0.5)$, since $g(0) = 0.5$. The model parameter vector $\boldsymbol{\theta}$ is usually estimated to maximize the likelihood of $P(y|\boldsymbol{x}; \boldsymbol{\theta})$ for training dataset $D = \{\boldsymbol{x}^{(m)}, y^{(m)}\}_{m=1}^M$ and the prior probability density of $\boldsymbol{\theta}$:

$$J_R(\boldsymbol{\theta}) = \sum_{m=1}^{M} \log P(y^{(m)}|\boldsymbol{x}^{(m)}; \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}). \quad (3)$$

In this paper, the classifier design approach that employs this training method is called LRM-L.

By contrast, in the training method proposed by (Jansche, 2005), the discriminative function $f(\boldsymbol{x}; \boldsymbol{w})$ is estimated to maximize the F_1-score of training dataset D. This training method employs an approximate form of the F_1-score obtained by using a logistic function.

The F_1-score is defined as $F_1 = 2(1/PR + 1/RE)^{-1}$, where PR and RE represent precision and recall defined as $PR = C/A$ and $RE = C/B$, respectively. Here, C represents the number of data samples whose true and predicted class label assignments, $y^{(n)}$ and $\hat{y}^{(n)}$, respectively, correspond to 1. A represents the number of data samples for which $\hat{y}^{(n)} = 1$. B represents the number of data samples for which $y^{(n)} = 1$. C, A, and B are computed for training dataset D as $C = \sum_{m=1}^M y^{(m)} \hat{y}^{(m)}$, $A = \sum_{m=1}^M \hat{y}^{(m)}$, and $B = \sum_{m=1}^M y^{(m)}$.

In (Jansche, 2005), $\hat{y}^{(m)}$ was approximated by using the discriminative and logistic functions shown in Eqs. (1) and (2) as

$$\hat{y}^{(m)} \approx g(\gamma f(\boldsymbol{x}^{(m)}; \boldsymbol{\theta})),\ \gamma > 0, \quad (4)$$

because $\lim_{\gamma \to \infty} g(\gamma f(\boldsymbol{x}^{(m)}; \boldsymbol{\theta})) = \hat{y}^{(m)}$. Then, an approximate distribution of the F_1-score for training dataset D was provided as

$$\tilde{F}_1(\boldsymbol{\theta}) = \frac{2 \sum_{m=1}^M g(\gamma f(\boldsymbol{x}; \boldsymbol{\theta})) y^{(m)}}{\sum_{m=1}^M y^{(m)} + \sum_{m=1}^M g(\gamma f(\boldsymbol{x}; \boldsymbol{\theta}))}. \quad (5)$$

The $\boldsymbol{\theta}$ estimate for the discriminative function $f(\boldsymbol{x}; \boldsymbol{\theta})$ can be computed to maximize $J_F(\boldsymbol{\theta}) = \log \tilde{F}_1(\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ around the initial $\boldsymbol{\theta}$ value by using a gradient method. In this paper, the classifier design approach that uses this training method is called LRM-F.

3 Proposed Method

We propose a framework for designing a multi-label classifier based on the combination of multiple models. In our formulation, multiple models are combined with weights estimated to maximize the F_1-scores of the training dataset. In this section, we show our formulation for model combination and training methods for combination weights.

3.1 Combination of Multiple Models for Multi-label Categorization

Multi-label categorization is the task of selecting multiple category labels from K pre-defined category labels for each data sample. Multi-label classifiers provide a map from a feature vector $\boldsymbol{x}$ to a category label assignment vector $\boldsymbol{y} = (y_1, \ldots, y_k, \ldots, y_K)^t$, where $y_k^{(n)} = 1$ $(= 0)$ indicates that the kth category label is assigned (unassigned) to $\boldsymbol{x}^{(n)}$.

In our formulation, we first design multiple models for binary classification per category and obtain $J \times K$ discriminative functions, where J is the number of models. The discriminative function of the jth model for the kth category is denoted by $f_{jk}(\boldsymbol{x}; \boldsymbol{\theta}_{jk})$, where $\boldsymbol{\theta}_{jk}$ represents the model parameter vector. Let $\Theta = \{\boldsymbol{\theta}_{jk}\}_{j,k}$ be a model parameter set. We train model parameter vectors individually with each model training algorithm and obtain the estimate $\hat{\Theta} = \{\hat{\boldsymbol{\theta}}_{jk}\}_{jk}$. Then, we define the discriminative function of our multi-label classifier by combining multiple models such as

$$f_k(\boldsymbol{x}; \hat{\Theta}, \boldsymbol{w}) = \sum_{j=1}^{J} w_j f_{jk}(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_{jk}) + w_0, \ \forall k, \quad (6)$$

where $\boldsymbol{w} = (w_0, w_1, \ldots, w_j, \ldots, w_J)^t$ is a weight parameter vector and is independent of k. w_j provides the combination weight of the jth model, and w_0 is the bias factor for adjusting the threshold of the category label assignment.

We estimate the $\boldsymbol{w}$ value to maximize the micro-averaged F_1-score (F_μ), which is often used for evaluating multi-label categorization performance. The F_μ-score of training dataset $D = \{\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\}_{m=1}^{M}$ is calculated as

$$F_\mu = \frac{2\sum_{m=1}^{M}\sum_{k=1}^{K} y_k^{(m)} \hat{y}_k^{(m)}}{\sum_{m=1}^{M}\sum_{k=1}^{K} y_k^{(m)} + \sum_{m=1}^{M}\sum_{k=1}^{K} \hat{y}_k^{(m)}}, \quad (7)$$

We provide an approximate form of the F_μ-score of the training dataset, $\tilde{F}_\mu(\hat{\Theta}, \boldsymbol{w})$, by using the approximation:

$$\hat{y}_k^{(m)} \approx g(\gamma f_k(\boldsymbol{x}^{(m)}; \hat{\Theta}, \boldsymbol{w})), \ \gamma > 0, \quad (8)$$

as shown in Eq. (4). In our proposed method, $\boldsymbol{w}$ is estimated to maximize $\tilde{F}_\mu(\hat{\Theta}, \boldsymbol{w})$.

However, training dataset D is also used to estimate Θ. Using the same training data samples for both Θ and $\boldsymbol{w}$ may lead to a bias estimation of $\boldsymbol{w}$. Thus, we used an n-fold cross-validation of the training data samples to estimate $\boldsymbol{w}$ as in (Wolpert, 1992). Let $\hat{\Theta}^{(-m)}$ be the model parameter set estimated by using $n-1$ training data subsets not containing $\{\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)}\}$. Then, using

$$\tilde{F}_\mu = \frac{2\sum_{m,k} y_k^{(m)} g(\gamma f_k(\boldsymbol{x}; \hat{\Theta}^{(-m)}, \boldsymbol{w}))}{\sum_{m,k} y_k^{(m)} + \sum_{m,k} g(\gamma f_k(\boldsymbol{x}; \hat{\Theta}^{(-m)}, \boldsymbol{w}))}, \quad (9)$$

we provide the objective function of $\boldsymbol{w}$ such that

$$J_\mu(\boldsymbol{w}) = \log \tilde{F}_\mu + \log p(\boldsymbol{w}), \quad (10)$$

where $p(\boldsymbol{w})$ is a prior probability density of $\boldsymbol{w}$. We use a Gaussian prior (Chen and Rosenfeld, 1999) with the form as $p(\boldsymbol{w}) \propto \prod_{j=0}^{J} \exp\{-(w_j - \rho_j)^2/2\sigma_j^2\}$, where σ_j, and ρ_j are hyperparameters in the Gaussian prior. We compute an estimate of $\boldsymbol{w}$ to maximize $J_\mu(\boldsymbol{w})$ around the initial $\boldsymbol{w}$ value by using a quasi-Newton method. In this paper, this formulation is called *model combination by micro-averaged F_1-score maximization* (MC-F_μ).

3.2 Other Training Methods

In multi-label categorization problems, the macro-averaged F_1-score (F_M) is also used to evaluate classifiers. Moreover, the average labeling F_1-score (F_L) has been used to evaluate the average labeling performance of classifiers for data samples (Kazawa et al., 2005). These F_1-scores are computed for training dataset D as

$$F_M = \frac{1}{K}\sum_{k=1}^{K} \frac{2\sum_{m=1}^{M} y_k^{(m)} \hat{y}_k^{(m)}}{\sum_{m=1}^{M} y_k^{(m)} + \sum_{m=1}^{M} \hat{y}_k^{(m)}}, \quad (11)$$

$$F_L = \frac{1}{M}\sum_{m=1}^{M} \frac{2\sum_{k=1}^{K} y_k^{(m)} \hat{y}_k^{(m)}}{\sum_{k=1}^{K} y_k^{(m)} + \sum_{k=1}^{K} \hat{y}_k^{(m)}}. \quad (12)$$

Using Eq. (8), we can also obtain the approximate forms, $\tilde{F}_M(\hat{\Theta}, \boldsymbol{w})$ and $\tilde{F}_L(\hat{\Theta}, \boldsymbol{w})$, of the F_M-

and F_L-scores, and then present similar objective functions to that for the F_μ-score. Therefore, in the next section, we examine experimentally the performance of classifiers obtained by estimating $\boldsymbol{w}$ to maximize $\tilde{F}_M(\hat{\Theta}, \boldsymbol{w})$ and $\tilde{F}_L(\hat{\Theta}, \boldsymbol{w})$. In this paper, these model combination methods based on F_M- and F_L-scores are called MC-F_M and MC-F_L, respectively.

4 Experiments

4.1 Test Collections

To evaluate our proposed method empirically, we used three test collections: Reuters-21578 (Reuters), WIPO-alpha (WIPO), and Japanese Patent (JPAT) datasets. Reuters and WIPO are English document datasets and have often been employed for benchmark tests of multi-label classifiers.

The Reuters dataset contains news articles from the Reuters newswire and consists of 135 topic categories. Following the setup in (Yang and Liu, 1999), we extracted 7770 and 3019 articles as training and test samples, respectively. A subset consisting of the training and test samples contained 90 topic categories. We removed vocabulary words included either in the stoplist or in only one article. There were 16365 vocabulary words in the dataset.

The WIPO dataset consists of patent documents categorized using the International Patent Classification (IPC) taxonomy (Fall et al., 2003). The IPC taxonomy has four hierarchical layers: *Section*, *Class*, *Subclass*, and *Group*. Using patent documents belonging to *Section D* (textiles; paper), we evaluated classifiers in a task that consisted of selecting assigned category labels from 160 groups for each patent document. Following the setting provided in the dataset, we extracted 1352 and 358 patent documents as training and test samples, respectively. We removed vocabulary words in the same way as for Reuters. There were 45895 vocabulary words in the dataset.

The JPAT dataset (Iwayama et al., 2007) consists of Japanese patent documents published between 1993 and 1999 by the Japanese Patent Office. These documents are categorized using a taxonomy consisting of *Themes* and *F-terms*. The themes are top-label categories, and the patent documents belonging to each theme are categorized by using F-terms. Using patent documents belonging to *Theme 5J104*, we evaluated classifiers in a task that consisted of selecting assigned category labels from 268 F-terms for each patent document. 1920 patent documents published between 1993 and 1997 were used as training samples, and 544 patent documents published between 1998 and 1999 were used test samples. We extracted Japanese nouns, verbs, and adjectives from patent documents by using a morphological analyzer named MeCab [1], and removed vocabulary words included in only one patent document. There were 21135 vocabulary words in the dataset.

	Reuters	WIPO	JPAT
N_{av}	1.17	1.28	10.5
N_{max}	15	6	40
K	90	160	268
N_{ds}	10789	1710	2464
N_{LC}	468	378	2430
N_{ds}/N_{LC}	23.1	4.52	1.01

Table 1: Statistical information of three datasets: N_{av} and N_{max} are the average and maximum number of assigned category labels per data sample, respectively. K and N_{ds} are the number of category labels and data samples, respectively. N_{LC} is the number of category label combinations appearing in each dataset.

Table 1 shows statistical information about the category label assignment of the data samples for the three datasets. The average numbers of assigned category labels per data sample, N_{av}, for Reuters and WIPO were close to 1 and much smaller than that for JPAT. The number of category label combinations, N_{LC}, included in JPAT was larger than those for Reuters and WIPO. These statistical information results show that JPAT is a more *complex* multi-label dataset than Reuters or WIPO.

4.2 Experimental Settings

For text categorization tasks, we employed word-frequency vectors of documents as feature vectors input into classifiers, using the independent word-based representation, known as the Bag-of-Words (BOW) representation. We normalized the L1-norms of the word-frequency vectors to 1, to mitigate the effect of vector size on computation. We did not employ any word weighting methods such as inverse document frequency (IDF).

[1] http://mecab.sourceforge.net/

We constructed three multi-label text classifiers based on our proposed model combination methods, MC-F_μ, MC-F_M, and MC-F_L, where LRM and SVM ($J = 2$) were employed as binary classification models combined with each method. We trained the LRM by using LRM-L described in Section 2, where a Gaussian prior was used as the prior probability density of the parameter vectors. We provided the SVM by using SVM^{light} [2] (SVM-L), where we employed a linear kernel function and tuned the C (penalty cost) parameter as a hyperparameter.

To evaluate our proposed method, we examined the micro- and macro-averaged, and average labeling F_1-scores (F_μ, F_M, and F_L), of test samples obtained with the three classifiers based on MC-F_μ, MC-F_M, and MC-F_L. We compared the performance of the three classifiers with that of two binary classification approaches, where LRM-L or SVM-L was used for binary classification.

We also examined two binary classification approaches using LRM-F and SVM-F. For LRM-F, we used a Gaussian prior and provided the initial parameter vector with a parameter estimate obtained with LRM-L. SVM-F is a binary classifier design approach that employs SVM^{perf} [3]. For SVM-F, we used a linear kernel function, set the L (loss parameter) parameter to maximize the F_1-score, and tuned the C (penalty cost) parameter as a hyperparameter.

Moreover, we examined the performance of the *Maximal Margin Labeling* (MML) method (Kazawa et al., 2005), which models the map from feature vectors to category label assignment vectors, because it was reported that MML provides better performance than binary classification approaches.

We tuned the hyperparameter of SVM-F for JPAT to provide good performance for test samples, because the computational cost for training was high. We tuned the other hyperparameters by using a 10-fold cross-validation of training samples.

Method	F_μ (P_μ/R_μ)	F_M (P_M/R_M)	F_L (P_L/R_M)
MC-F_μ	87.0 (87.4/86.7)	51.3 (60.0/48.4)	90.0 (90.1/92.3)
MC-F_M	85.0 (80.8/89.5)	53.9 (54.9/58.4)	89.7 (88.5/94.1)
MC-F_L	86.3 (84.3/88.3)	53.4 (59.6/52.6)	90.0 (89.3/93.6)
LRM-L	85.2 (87.3/83.2)	46.1 (55.0/43.1)	86.9 (87.6/88.6)
LRM-F	85.2 (87.2/83.2)	47.4 (58.5/42.7)	87.0 (87.6/88.7)
SVM-L	87.1 (92.9/82.0)	48.9 (58.9/45.8)	88.1 (89.3/88.8)
SVM-F	82.4 (78.9/86.2)	51.4 (49.4/60.1)	87.4 (86.9/91.4)
MML	**87.8** (92.6/83.4)	**59.3** (62.6/60.0)	**91.2** (91.7/93.2)

(a) Reuters

Method	F_μ (P_μ/R_μ)	F_M (P_M/R_M)	F_L (P_L/R_M)
MC-F_μ	**51.4** (57.3/46.6)	30.4 (35.8/30.3)	46.9 (48.3/51.5)
MC-F_M	48.1 (46.1/50.4)	32.2 (33.8/36.0)	46.8 (46.3/56.0)
MC-F_L	48.6 (45.8/51.9)	**32.5** (33.4/36.5)	47.1 (46.4/56.8)
LRM-L	40.5 (68.0/28.9)	22.1 (33.7/17.9)	32.7 (36.5/32.0)
LRM-F	41.0 (68.6/29.2)	22.3 (34.0/18.1)	33.2 (37.0/32.4)
SVM-L	41.8 (61.9/31.5)	24.4 (34.2/21.0)	35.1 (38.8/35.3)
SVM-F	48.3 (53.8/43.8)	32.3 (37.4/31.8)	45.6 (47.9/49.6)
MML	48.6 (54.9/43.6)	30.8 (36.5/29.7)	**49.4** (56.2/48.4)

(b) WIPO

Method	F_μ (P_μ/R_μ)	F_M (P_M/R_M)	F_L (P_L/R_M)
MC-F_μ	41.8 (42.6/41.1)	17.5 (21.4/17.4)	40.2 (43.5/44.4)
MC-F_M	40.6 (35.8/46.7)	**20.2** (20.4/23.1)	39.4 (37.7/50.6)
MC-F_L	**42.1** (42.3/41.9)	17.6 (21.1/17.8)	**40.5** (43.2/45.2)
LRM-L	33.9 (44.4/27.4)	15.8 (20.9/14.0)	32.2 (46.5/29.9)
LRM-F	36.9 (44.6/31.5)	16.9 (22.9/14.7)	35.1 (47.3/34.1)
SVM-L	33.3 (39.6/28.7)	16.3 (20.9/14.6)	31.9 (42.4/31.6)
SVM-F	32.2 (28.6/36.8)	19.7 (15.0/38.4)	31.0 (30.7/40.0)
MML	32.7 (42.1/26.8)	14.7 (19.4/12.9)	32.2 (51.8/30.5)

(c) JPAT

Table 2: Micro- and macro-averaged, and average labeling F_1-scores (%) with our proposed and conventional methods.

4.3 Results and Discussion

In Table 2, we show the classification performance obtained for three datasets with our proposed and other methods described in Section 4.2. We examined nine evaluation scores: the micro-averaged F_1-score (F_μ), precision (P_μ), and recall (R_μ), the macro-averaged F_1-score (F_M), precision (P_M), and recall (R_M), and the average labeling F_1-score (F_L), precision (P_L), and recall (R_L) of the test samples. F_M and P_M were calculated by regarding both the F_1-score and precision as zero for the categories where there were no data samples predicted as positive samples.

LRM-F and SVM-F outperformed LRM-L and SVM-L in terms of F_M-score for the three datasets, respectively. The training methods of LRM-F and SVM-F were useful to improve the F_M-scores of LRM and SVM, as reported in (Jansche, 2005; Joachims, 2005). The F_μ- and F_L-scores of LRM-F were similar or better than those of LRM-L. LRM-F was effective in improving not only the F_M-score but also the F_μ- and F_L-scores obtained with LRM.

Let us evaluate our model combination methods.

[2]http://svmlight.joachims.org/

[3]http://svmlight.joachims.org/svm_perf.html

MC-F_μ provided better F_μ-scores than LRM-F and SVM-F. The F_M-scores of MC-F_M were similar or better than those of LRM-F and SVM-F. Moreover, MC-F_L outperformed LRM-F and SVM-F in terms of F_L-scores. The binary classifiers designed by using LRM-F and SVM-F were trained to maximize the F_1-score for each category. On the other hand, MC-F_μ, MC-F_M, and MC-F_L classifiers were constructed by combining LRM and SVM with weights estimated to maximize the F_μ-, F_M-, and F_L-scores, respectively. The experimental results show that our training methods for combination weights were useful for obtaining better multi-label classifiers.

MC-F_μ, MC-F_M, and MC-F_L outperformed MML as regards the three F_1-scores for JPAT. However, MML performed better for Reuters than MC-F_μ, MC-F_M, and MC-F_L, and provided a better F_L-score for WIPO. As shown in Table 1, there were more category label combinations for JPAT than for Reuters or WIPO. As a result, there were fewer data samples for the same category label assignment for JPAT. Therefore, MML, which learns the map directly from the feature vectors to the category label assignment vectors, would have been overfitted to the training dataset for JPAT. By contrast, our model combination methods employ binary classifiers for each category, which mitigates such an overfitting problem. Our model combination methods will be useful for *complex* datasets where there are many category label combinations.

5 Conclusion

We proposed a multi-label classifier design method based on model combination. The main idea behind our proposed method is to combine multiple models with weights estimated to maximize evaluation scores such as the micro- and macro-averaged, and average labeling F_1-scores. Using three real text datasets, we confirmed experimentally that our proposed method provided similar or better performance than conventional binary classification approaches to multi-label categorization. We also confirmed that our proposed method was useful for datasets where there were many combinations of category labels. Future work will involve training our multi-label classifier by using labeled and unlabeled samples, which are data samples with and without category label assignment.

References

David A. Bell, J. W. Guan, and Yaxin Bi. 2005. On combining classifier mass functions for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1307–1319.

Stanley F. Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University.

C. J. Fall, A. Törcsvári, K. Benzineb, and G. Karetka. 2003. Automated categorization in the international patent classification. *ACM SIGIR Forum*, 37(1):10–25.

Giorgio Fumera and Fabio Roli. 2005. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942–956.

Zoubin Ghahramani and Hyun-Chul Kim. 2003. Bayesian classifier combination. Technical report, Gatsby Computational Neuroscience Unit, University College London.

Makoto Iwayama, Atsushi Fujii, and Noriko Kando. 2007. Overview of classification subtask at NTCIR-6 patent retrieval task. In *Proceedings of the 6th NTCIR Workshop Meeting on Evaluation of Information Access Technologies (NTCIR-6)*, pages 366–372.

Martin Jansche. 2005. Maximum expected F-measure training of logistic regression models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP2005)*, pages 692–699.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML '98)*, pages 137–142.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, pages 377–384.

Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. 2005. Maximal margin labeling for multitopic text categorization. In *Advances in Neural Information Processing Systems 17*, pages 649–656. MIT Press, Cambridge, MA.

Leah S. Larkey and W. Bruce Croft. 1996. Combining classifiers in text categorization. In *Proceedings of the 19th ACM International Conference on Research and Development in Information Retrieval (SIGIR-96)*, pages 289–297.

Kai Ming Ting and Ian H. Witten. 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289.

David H. Wolpert. 1992. Stacked generalization. *Newral Networks*, 5(2):241–259.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR-99)*, pages 42–49.

An Experimental Comparison of the Voted Perceptron and Support Vector Machines in Japanese Analysis Tasks

Manabu Sassano
Yahoo Japan Corporation
6-10-1 Roppongi, Minato-ku, Tokyo 106-6182 Japan
msassano@yahoo-corp.jp

Abstract

We examine various aspects of the voted perceptron and support vector machines in classification tasks in NLP rather than ranking tasks. These aspects include training time, accuracy and learning curves. We used Japanese dependency parsing as a main task for experiments, and Japanese word segmentation and bunsetsu (base phrase in Japanese) chunking as auxiliary tasks. In our experiments we have observed that the voted perceptron is comparable to SVM in terms of accuracy and, in addition, as to learning time and prediction speed the voted perceptron is considerably better than SVM.

1 Introduction

Support vector machines (SVM) (Vapnik, 1995) have been shown to be effective for many natural language processing (NLP) tasks (e.g., (Kudo and Matsumoto, 2001; Kudo and Matsumoto, 2002)). However, there are still some practical difficulties when we apply SVM to NLP tasks. The weakness of SVM is that they are not easy to implement and their learning process is slow, especially with polynomial kernels.

(Freund and Schapire, 1999) propose the voted perceptron, which is an improved version of Perceptron (Rosenblatt, 1958), and they give theoretical analysis and have proved a good performance for the hand-written digit recognition. Although Collins and his colleagues use the voted perceptron for ranking in various NLP tasks (Collins, 2002b; Collins and Duffy, 2002; Collins, 2002a) and obtain impressive results, the use as a classifier has been not sufficiently examined. In particular, it would be an interesting question whether or not the voted perceptron is comparable to SVM for NLP tasks that are formalized as a classification one.

In this paper we focus on comparison of SVM and the voted perceptron to investigate the usefulness of the voted perceptron in NLP tasks. We would like to know the strength and weakness of the voted perceptron. We choose three tasks for this purpose. These tasks are Japanese word segmentation, *bunsetsu* (base phrase in Japanese) chunking, and dependency parsing.

Experiments indicate that SVM and the voted perceptron are equally good for the three tasks in terms of accuracy. However, the voted perceptron is superior to SVM in terms of learning time, prediction time, and memory footprint.

2 The Voted Perceptron

Following (Freund and Schapire, 1999), we show the training and prediction algorithm of the voted perceptron in Figure 1. The voted perceptron as well as SVM can use a kernel function. We show in Figure 2 the algorithm of the voted perceptron with a kernel function. This algorithm seems to require $O(k^2)$ kernel calculations. However, we can avoid them by taking advantage of the recurrence $\boldsymbol{v}_{j+1} \cdot \boldsymbol{x} = \boldsymbol{v}_j \cdot \boldsymbol{x} + y_{u_j} K(\boldsymbol{x}_{u_j}, \boldsymbol{x})$.[1]

3 Task Description

We used Japanese dependency parsing as a main task for experiments and Japanese word segmenta-

[1] Herbrich describes an optimized version of the algorithm of the kernel perceptron (Herbrich, 2002, page 322). We can use the same technique in training of the kernel version of the voted perceptron.

Training

Input: a labeled training set:
$\langle(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\rangle$.
number of epochs: T

Output: a list of weighted perceptrons:
$\langle(\boldsymbol{v}_1, c_1), \ldots, (\boldsymbol{v}_k, c_k)\rangle$

- Initialize: $k := 0, \boldsymbol{v}_1 := \mathbf{0}, c_1 := 0$.
- Repeat T times:
  - For $i := 1, \ldots, m$:
    - Compute prediction: $\hat{y} :=$ sign$(\boldsymbol{v}_k \cdot \boldsymbol{x}_i)$
    - If $\hat{y} = y$ then $c_k := c_k + 1$.
      else $\boldsymbol{v}_{k+1} := \boldsymbol{v}_k + y_i \boldsymbol{x}_i$;
      $c_{k+1} := 1; k := k + 1$.

Prediction

Given: the list of weighted perceptrons:
$\langle(\boldsymbol{v}_1, c_1), \cdots, (\boldsymbol{v}_k, c_k)\rangle$
an unlabeled instance: $\boldsymbol{x}$

compute a predicted label $\hat{y}$ as follows:

$$s := \sum_{i=1}^{k} c_i \text{ sign}(\boldsymbol{v}_i \cdot \boldsymbol{x}); \hat{y} := \text{sign}(s).$$

Figure 1: The voted-perceptron algorithm

Training

Input: $\langle(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\rangle$ and T

Output: a list of mistaken examples and weights:
$\langle(u_1, c_1), \ldots, (u_k, c_k)\rangle$

- Initialize: $k := 0, \boldsymbol{v}_1 := \mathbf{0}, c_1 := 0$.
- Repeat T times:
  - For $i := 1, \ldots, m$:
    - Compute prediction:

$$\boldsymbol{v}_k \cdot \boldsymbol{x}_i := \sum_{j=1}^{k} y_{u_j} K(\boldsymbol{x}_{u_j}, \boldsymbol{x}_i); \hat{y} := \text{sign}(\boldsymbol{v}_k \cdot \boldsymbol{x}_i)$$

- If $\hat{y} = y$ then $c_k := c_k + 1$.
  else $u_{k+1} := i$;
  $c_{k+1} := 1; k := k + 1$.

Prediction

Given: $\langle(u_1, c_1), \cdots, (u_k, c_k)\rangle$ and $\boldsymbol{x}$

compute a predicted label $\hat{y}$ as follows:

$$\boldsymbol{v}_i \cdot \boldsymbol{x} := \sum_{j=1}^{i} y_{u_j} K(\boldsymbol{x}_{u_j}, \boldsymbol{x}); s := \sum_{i=1}^{k} c_i \text{ sign}(\boldsymbol{v}_i \cdot \boldsymbol{x});$$

$$\hat{y} := \text{sign}(s).$$

Figure 2: Algorithm of the voted-perceptron with a kernel function

tion and bunsetsu chunking as auxiliary tasks.

3.1 Japanese Dependency Parsing (JDP)

Japanese dependency parsing is to determine the dependency structure of a given sentence which is represented as a sequence of *bunsetsu*s (base phrases in Japanese). We employ the Stack Dependency Analysis (SDA) algorithm (Sassano, 2004; Nivre, 2003), which is very simple and easy to implement. Sassano (2004) has proved its efficiency in terms of time complexity and reported the best accuracy on the Kyoto University Corpus Version 2 (Kurohashi and Nagao, 1998). This algorithm, which can be used with any classifier that determines whether a given bunsetsu modifies another, is suitable for our study since we intend to test both SVM and the voted perceptron.

We use a set of standard features for this task. By the "standard features" here we mean the feature set commonly used in (Uchimoto et al., 1999; Sekine et al., 2000; Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2002; Sassano, 2004). We employ the features below for each bunsetsu:

1. Rightmost Content Word - major POS, minor POS, conjugation type, conjugation form, surface form (lexicalized form)
2. Rightmost Function Word - major POS, minor POS, conjugation type, conjugation form, surface form (lexicalized form)
3. Punctuation (periods, and commas)
4. Open parentheses and close parentheses
5. Location - at the beginning of the sentence or at the end of the sentence.

In addition, features as to the gap between two bunsetsus are also used. They include: distance, particles, parentheses, and punctuation.

3.2 Japanese Bunsetsu Chunking (JBC)

Following (Ramshaw and Marcus, 1995), we encode bunsetsu chunking as a tagging problem. In bunsetsu chunking, we use the chunk tag set {`B`, `I`} where `B` marks the first word of some bunsetsu and words marked `I` are inside a bunsetsu.

In our experiments on bunsetsu chunking, we estimated the chunk tag of each word using five

Table 1: The Size of the Training Data

	JWS	JBC	JDP
# of features	11916	121081	40842
# of examples	350584	198514	98689

words and their derived attributes. These five words are the word to be estimated and its two preceding/following words. Features are extracted from the followings for each word: word (token) itself, major POS, minor POS, conjugation type, conjugation form, the leftmost character, the character type of the leftmost character, the rightmost character, and the character type of the rightmost character. A character type has a value which indicates a script. This value can be one of the following: kanji (Chinese character), hiragana (Japanese syllabic character), katakana (another syllabic character), number, Latin letter, or symbol.

3.3 Japanese Word Segmentation (JWS)

Japanese word segmentation can be formulated as a classification task (Shinnou, 2000). Let a Japanese character sequence be $s = c_1 c_2 \cdots c_m$ and a boundary b_i exist between c_{i-1} and c_i. The b_i is either $+1$ (word boundary) or -1 (non-boundary). The word segmentation task can be defined as determining the class of the b_i.

We assume that each character c_i has two attributes. The first attribute is a character type (t_i). The second one is a character code (k_i). We use here five characters to decide a word boundary. A set of the attributes of $c_{i-2}, c_{i-1}, c_i, c_{i+1}$, and c_{i+2} is used to predict the label of the b_i.

4 Experimental Results and Discussion

4.1 Corpus

We used the Kyoto University Corpus Version 2 (Kurohashi and Nagao, 1998). Analysis systems used in any of our experiments were trained on the articles on January 1st through 8th (7,958 sentences) and tested on the articles on January 9th (1,246 sentences). The articles on January 10th were used for development. The usage of these articles is the same as in (Uchimoto et al., 1999; Sekine et al., 2000; Kudo and Matsumoto, 2002; Sassano, 2004). The size of the training data set is given in Table 1.

4.2 Parameters

We selected the best value of the misclassification cost C of SVM by using the development test set. We carried out training SVM with 0.0001, 0.001, 0.01, 0.1, and 1 as a value of C and measured accuracy on the development test set. We then used models with these best values of C on the test set. Similarly, as to the best value of the number of epoch T of the voted perceptron, we applied the same procedure and found the best value of T for the development test set.

We use polynomial kernels with the degree of 3 for all the experiments. The main reason for this is as follows. Polynomial kernels with the degree of 3 have been widely used for Japanese analysis tasks and they have reported better performance than that of other kernels. A cubic kernel would be a good first choice. In particular, cubic kernels are used for Japanese dependency parsing in many papers (e.g., (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2002; Sassano, 2004)). Thus there are additional benefits that we can compare our results with others.

4.3 Accuracy

We show in Table 2 the summary of performance for the three tasks with p-values of McNemar's test (Gillick and Cox, 1989) at .05 significance level. SVM outperforms the voted perceptron[2] in all the three tasks. In the cases of both JWS and JBC, the differences between SVM and the voted perceptron are statistically significant. However, in the case of JDP the difference is not significant. Since absolute differences are little in any of the cases, there would be no serious impact in many practical applications whichever you may choose.

Figures 3, 4 and 5 show how accuracy changes at each epoch. In the cases of JWS and JBC, the accuracy peaks at the epoch of around 15. On the other hand, in the case of JDP, the accuracy peaks at the epoch of 2 and then it fluctuates a little and declines gradually.

[2]We actually used "averaging" instead of voting for simplicity. That is, we used in Figure 2 $s = \sum_{i=1}^{k} c_i\,(\boldsymbol{v}_i \cdot \boldsymbol{x})$ instead of $s = \sum_{i=1}^{k} c_i\,\mathrm{sign}(\boldsymbol{v}_i \cdot \boldsymbol{x})$.

Table 2: Performance Summary

	JWS	JBC	JDP
SVM	**98.48%**	**99.69%**	**88.72%**
	$(C = 0.01)$	$(C = 1)$	$(C = 0.001)$
VP	98.32%	99.63%	88.48%
	$(T = 30)$	$(T = 6)$	$(T = 12)$
p-value	1.49e-06	0.021	0.12

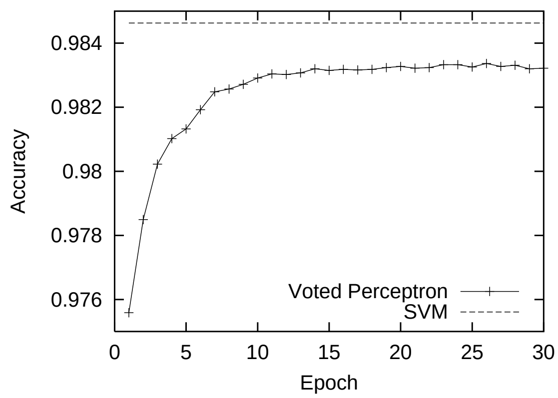


Figure 3: Accuracy of Japanese Word Segmentation

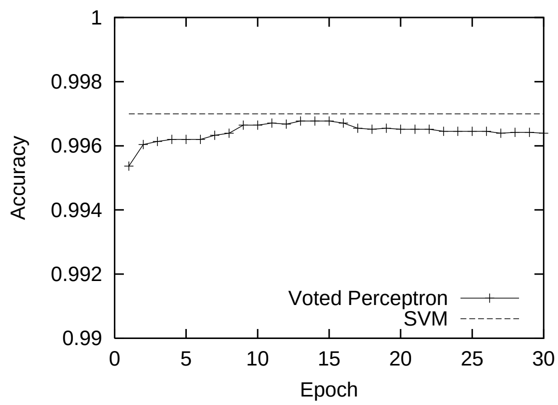


Figure 4: Accuracy of Bunsetsu Chunking

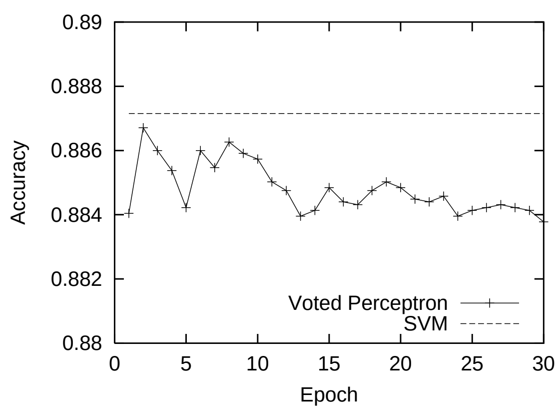


Figure 5: Accuracy of Dependency Analysis

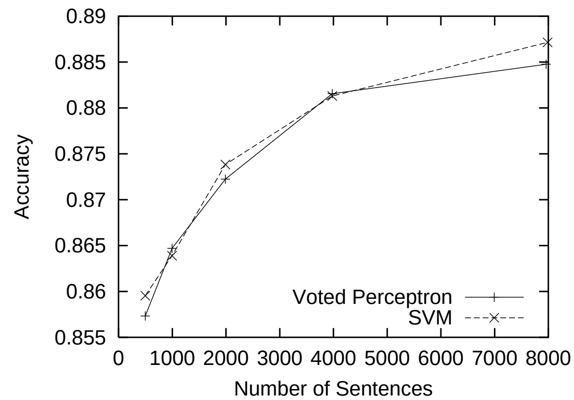


Figure 6: Learning Curves of Dependency Analysis

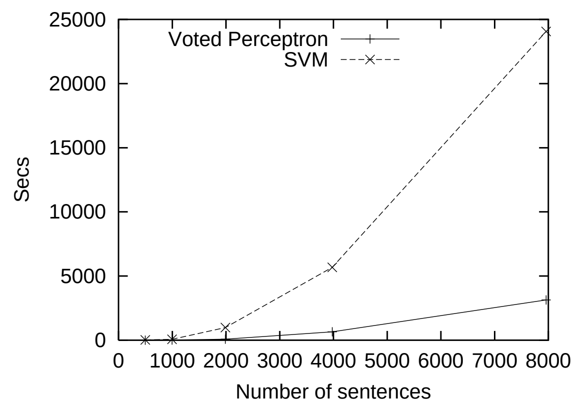


Figure 7: Learning Time

4.4 Learning Curves

We here show the learning curves (Figure 6) of SVM and the voted perceptron for the JDP task. Both curves exhibit a similar shape.

4.5 Learning Time

Now let us see the learning time of SVM and the voted perceptron. We examined the learning time[3] (Figure 7) of the JDP task.

We used LIBSVM (Chang and Lin, 2001) for SVM and an original tool written in C++ for the voted perceptron. LIBSVM used 300MB memory for kernel caching, while our tool for the voted perceptron used no extra memory. The learning time of the voted perceptron is more than five times faster than that of SVM although the tool for the voted perceptron requires less memory.

[3]Executed on FreeBSD with Pentium III 1.4GHz and 3GB memory.

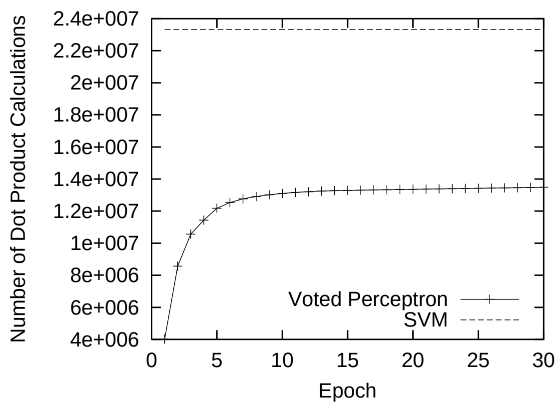


Figure 8: Number of dot product calculations in the training phase of JDP with 498 sentences.

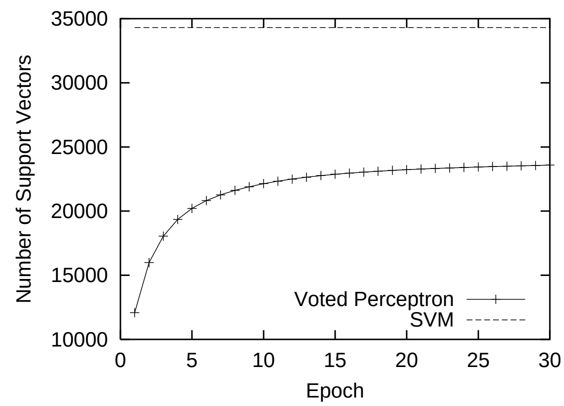


Figure 9: Number of Support Vectors

4.6 Number of Dot Product Calculations

Learning time somewhat may be affected by the implementation details of these tools. Therefore, we counted the number of dot product computation, which directly indicates the learning cost. Figure 8 shows the number of dot product calculations in the case of training of JDP with 498 sentences. The voted perceptron requires considerably fewer calculations of dot products than SVM does. This means the learning of the voted perceptron can be much faster than that of SVM.

4.7 Number of Support Vectors

We also measured the number of support vectors in models of both SVM and the voted perceptron[4]. Figure 9 shows the change of support vectors of the voted perceptron in the case of JDP with the full training data depending on the number of epochs. As (Freund and Schapire, 1999) pointed out, the number of support vectors of the voted perceptron is significantly fewer than that of SVM. This leads to faster prediction of the voted perceptron.

5 Related Work

Carreras et al. (2003) uses a voted perceptron for named entity recognition (NER). However, they have not compared their results with systems using SVMs. Therefore, it is not clear that the NER system with the voted perceptron has any advantages over NER systems with SVM.

Collins' work (Collins, 2002b; Collins and Duffy, 2002; Collins, 2002a) on the voted perceptron focuses mainly on ranking tasks in various problems. It does not treat classification tasks directly and there is no comparison with SVM.

6 Conclusion

In this paper we have compared SVM with the voted perceptron in three tasks of Japanese analysis. In our experiments we have observed that the voted perceptron is comparable to SVM in terms of accuracy and, in addition, as to learning time and prediction speed the voted perceptron is considerably better than SVM. These observations are consistent with the theoretical analysis and experimental results in (Freund and Schapire, 1999).

The voted perceptron is found to be a strong alternative to SVM in classification tasks in NLP as well as ranking tasks. If you choose SVM eventually, the voted perceptron would be very useful when designing a kernel because the same kernel function can be used in both SVM and the voted perceptron and you can obtain benefits from the easiness of implementation and the learning speed of the voted perceptron.

We have a plan to apply the voted perceptron to text classification and other diverse tasks in NLP. We would like to report experimental results and clear the effectiveness and the weakness of the voted perceptron.

[4]We use the term "support vectors" for the voted perceptron as well as SVM. "Support vectors" of the voted perceptron means vectors which are selected in the training phase and contribute to the prediction. Note that the number of support vectors in Figure 9 is fewer than the number of prediction mistakes in training of the voted perceptron.

References

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2003. Learning a perceptron-based named entity chunker via online recognition feedback. In *Proc. of the CoNLL 2003*, pages 156 – 159.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc. of ACL-2002*, pages 263–270.

Michael Collins. 2002a. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP-2002*, pages 1–8.

Michael Collins. 2002b. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proc. of ACL-2002*, pages 489–496.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277 – 296.

L. Gillick and Stephen Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. of ICASSP-89*, volume 1, pages 532 – 535.

Ralf Herbrich. 2002. *Learning Kernel Classifiers*. The MIT Press.

Taku Kudo and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proc. of EMNLP/VLC 2000*, pages 18–25.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proc. of NAACL 2001*, pages 192–199.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL-2002*, pages 63–69.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proc. of the 1st LREC*, pages 719–724.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT-03*, pages 149–160.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proc. of VLC 1995*, pages 82–94.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407.

Manabu Sassano. 2004. Linear-time dependency analysis for Japanese. In *Proc. of COLING 2004*, pages 8–14.

Satoshi Sekine, Kiyotaka Uchimoto, and Hitoshi Isahara. 2000. Backward beam search algorithm for dependency analysis of Japanese. In *Proc. of COLING-00*, pages 754–760.

Hiroyuki Shinnou. 2000. Deterministic Japanese word segmentation by decision list method. In *Proc. of PRICAI-2000*, page 822.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proc. of EACL-99*, pages 196–203.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.

Learning Decision Lists with Known Rules for Text Mining

Venkatesan Chakravarthy
IBM India Research Lab
vechakra@in.ibm.com

Sachindra Joshi
IBM India Research Lab
jsachind@in.ibm.com

Ganesh Ramakrishnan
IBM India Research Lab
ganramkr@in.ibm.com

Shantanu Godbole
IBM India Research Lab
shgodbol@in.ibm.com

Sreeram Balakrishnan
IBM Silicon Valley Lab
sreevb@us.ibm.com

Abstract

Many real-world systems for handling unstructured text data are rule-based. Examples of such systems are named entity annotators, information extraction systems, and text classifiers. In each of these applications, ordering rules into a decision list is an important issue. In this paper, we assume that a set of rules is given and study the problem (MaxDL) of ordering them into an optimal decision list with respect to a given training set. We formalize this problem and show that it is NP-Hard and cannot be approximated within any reasonable factors. We then propose some heuristic algorithms and conduct exhaustive experiments to evaluate their performance. In our experiments we also observe performance improvement over an existing decision list learning algorithm, by merely re-ordering the rules output by it.

1 Introduction

Rule-based systems have been extensively used for several problems in text mining. Some problems in text mining where rule-based systems have been successfully used are part of speech tagging (Brill, 1992), named entity annotation (Grishman, 1997; Appelt et al., 1995), information extraction (Maynard et al., 2001), question answering (Riloff and Thelen, 2000) and classification (Han et al., 2003; Li and Yamanishi, 1999; Sasaki and Kita, 1998). Several studies have been conducted that compare the performance of rule-based systems and other machine learning techniques with mixed results. While there is no clear winner between the two approaches in terms of performance, the rule-based approach is clearly preferred in operational settings (Borthwick, 1999; Varadarajan et al., 2002). Rule-based systems are human comprehensible and can be improved over time. Therefore, it is imperative to develop methods that assist in building rule-based systems.

A rule-based system consists of a set of rules. These rules can either be manually designed or could be learnt from a training set using rule-induction techniques (J. and G, 1994; Cohen, 1995). Each rule consists of an *antecedent* or *pattern* and a *consequent* or *predicted annotation*. In this paper, we will restrict our attention to a broad class of rules in which the antecedent describes a series of conditions on the input item and the consequent specifies the label that applies to instances *covered* by the antecedent. The conditions could also be expressed as patterns in regular or more powerful grammars.

In general, rules could be ambiguous, *i.e.*, multiple rules could *cover* an instance. A common approach for resolving this ambiguity is to define an ordering on the rules (Maynard et al., 2001; Borthwick, 1999). A decision list is one such mechanism (Rivest, 1987). A set of rules that are intended to be interpreted in a sequence is called a *decision list*. In other words, a decision list is an ordering of the given set of rules. Given an instance t, the rules are applied in the specified order until a pattern of a rule R covers t. The instance t is assigned the predicted annotation associated with R.

In this paper, we study the problem of arranging a given set of rules into the “best” decision list. Learning decision lists using training data has been studied in the past (Rivest, 1987; J. and G, 1994; Cohen, 1995; Li and Yamanishi, 1999). These methods attempt to simultaneously learn rules and their ordering. Typically they use *separate and conquer* (Witten and Frank, 2005) strategy and order generated rules as they are discovered. The generation and ordering of rules are not considered as two separate

tasks. In contrast, we assume that the rules are given to us and study the problem of arranging them into an optimal decision list, where optimality is determined over a training data set. Our approach is motivated by the observation that in many operational settings, it is easier and preferred to get a set of rules designed by domain experts (Lewis et al., 2003). Alternatively, the set of rules can be determined using existing techniques for rule learning (J. and G, 1994; Cohen, 1995; Califf and Mooney, 1998). The separation of rule ordering from rule generation allows us to analyze the problem of ordering in detail and to develop *effective* methods for rule ordering. We demonstrate the usefulness of the proposed methods for ordering manually designed rules in the task of named entity annotation and machine learnt rules in the task of classification.

We determine the ordering of the given set of rules based on a training set. A training set consists of a set of pairs (t_i, a_i) where t_i is an instance and a_i is its actual annotation. Given a set of rules and a training data set, we define the problem as follows: *Arrange the rules into a decision list such that maximum number of instances are assigned the correct annotation.* We refer to this problem as the MAXDL problem. We show that this problem is NP hard and cannot approximated within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$. We then propose some heuristics and present an experimental study of these heuristics. Our experimental results show performance improvement over an existing decision list learning algorithm, by merely reordering the rules output by that algorithm. We also illustrate the performance improvements obtained by applying our algorithms for ordering named entity annotation rules and classification rules.

In the rest of the paper we formalize the MAXDL problem (§2), show it is NP-hard and can't be approximated within reasonable factors (§3), and propose heuristics in a greedy framework (§4). We present experiments (§5) and conclude with Section§6.

2 MAXDL Problem Definition and Notations

The input consists of a set of *instances* $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$, a set of annotations $\mathcal{A}$ and a set of rules $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$. Each rule $R_i = (p, a)$ is a pair, where p is called the *pattern* and $a \in \mathcal{A}$ is called the *predicted annotation*. The patten p will be given as a set $p \subseteq \mathcal{I}$; we say that the instances in p are *covered* by R. The input also includes a mapping $A : \mathcal{T} \mapsto \mathcal{A}$, that provides for each instance t an annotation $A(t)$, called the *actual annotation* of t. The pair $(\mathcal{T}, A)$ is the *training data*.

Given the above input, a *decision list* L is an ordering (i.e. permutation) of the input rules. The list L assigns an annotation to each instance t as defined below. We consider each rule according to the ordering given by L until we find a rule $R_i = (p, a)$ that covers t and assign the annotation a to t. We denote by $L(t)$ the annotation assigned by L to t. Thus, L defines a function $L : \mathcal{I} \mapsto \mathcal{A}$. We say that the list L *correctly annotates* an instance t, if the annotation assigned by L matches the actual annotation of t, i.e., $L(t) = A(t)$.

Given the above input, the MAXDL problem is to to construct a decision list L such that the number of instances correctly annotated by L, is maximized i.e., we want to maximize $|\{t|A(t) = L(t)\}|$.

Notations:

Let $R = (p, a)$ be a rule and t be an instance covered by R. We say that a rule R *correctly covers* t, if $a = A(t)$. Similarly, R said to *incorrectly cover* t, if $a \neq A(t)$.

Let L be a decision list. We say that an instance t is *happy* under L, if L correctly annotates t, i.e., $L(t) = A(t)$. Let $\text{Happy}(L)$ denote the set of instances that are happy under L. Notice that the MAXDL problem asks for a decision list L such that $|\text{Happy}(L)|$ is maximized.

3 NP-Hardness and Inapproximability

In this section, we prove that the MAXDL problem is NP-Hard and also show that the problem cannot even be approximated with any constant factor.

Theorem 1 *The* MAXDL *problem is* NP*-Hard.*

Proof: We give a reduction from the maximum independent set problem (MAXIS), a well-known NP-Hard problem (Garey and Johnson, 1979). Recall that an independent set in a graph refers to any subset of vertices such that no two vertices from the set share an edge. The MAXIS problem is to find the largest independent set in a given undirected graph.

Let $G = (V, E)$ be the input graph having vertex set $V = \{v_1, v_2, \ldots, v_n\}$. We create an instance of the MAXDL problem as follows. For each vertex v_i, we add an annotation a_i to $\mathcal{A}$, an instance t_i to $\mathcal{T}$ and a rule R_i to $\mathcal{R}$. We declare a_i to be the actual annotation of t_i. The predicted annotation of R_i is set to a_i. We define R_i to cover only the instance t_i and the instances corresponding to the neighbors of v_i. Meaning, R_i covers the instances in the set $\{t_i\} \cup \{t_j | (v_i, v_j) \in E\}$. This completes the reduction. We claim that given a decision list L having k happy instances, we can construct an independent set of size k and vice versa. The NP-Hardness of MAXDL follows from the claim. We now proceed to prove the claim.

Consider a decision list L. Notice that for any instance t_i, R_i is the only rule that correctly covers t_i. Take any two different instances t_i and t_j that are happy under L. Without loss of generality, assume that R_i appears before R_j in L. Now, if R_i covers t_j, t_j would be unhappy under L. So, R_i does not cover t_j, which implies that v_j is not a neighbor of v_i (i.e., $(v_i, v_j) \notin E$). Hence, the set $I = \{v_i | t_i \in \mathrm{Happy}(L)\}$ is an independent set of G. We note that $|I| = |\mathrm{Happy}(L)|$.

Conversely, consider an independent set I of G. Let $R(I) = \{R_i | v_i \in I\}$. Form a decision list L by first arranging the rules from $R(I)$ in any arbitrary order followed by arranging the rest of rules in any arbitray order. Notice that for any vertex $v_i \in I$, R_i correctly covers t_i and no other rule appearing before R_i covers t_i. Thus, t_i is happy under L. It follows that $|\mathrm{Happy}(L)| \geq |I|$. We have proved that the MAXDL problem is NP-Hard. □

In our NP-Hardness reduction, we had shown that given a decision list L, we can construct an independent set I such that $|\mathrm{Happy}(L)| = |I|$, and vice versa. This means that any approximation algorithm for the MAXDL problem can be translated (by combining it with our NP-Hardness reduction) into an equally good approximation algorithm for the MAXIS problem. Corollary 1 follows from (Zuckerman, 2006).

Corollary 1 *If* NP $\neq$ P *then for any* $\epsilon > 0$, *the* MAXDL *problem cannot approximated within a factor of* $n^{1-\epsilon}$. *In particular, the problem is not approximable within any constant factor.*

4 Heuristic Algorithms for the MAXDL Problem

As the MAXDL problem is hard to approximate, we turn to heuristic approaches. All our heuristics fall into a natural greedy framework, described below.

4.1 A Greedy Framework

Our greedy framework for finding a decision list is as follows. In each iteration we greedily choose a rule and output it. For this purpose, we use some scoring function for assigning scores to the rules and choose the rule having the maximum score. Then the chosen rule is deleted. The process is continued until all the rules are output. The above procedure gives us a decision list. We present this general framework in the Figure 1. The only unspecified part in the above framework is the scoring function. Intuitively, the scoring function tries to measure the goodness of a rule.

Given rule set $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$, instance set $\mathcal{T}$ and the actual annotations $A(\cdot)$
while $\mathcal{R} \neq null$ **do**
  (re)compute scores for each rule in R, based on the scoring function
  select the rule R that has the maximum score
  remove R from the set $\mathcal{R}$
  remove from $\mathcal{T}$ all the instances covered by R
end while

Figure 1: A Greedy Framework for MAXDL problem

For a rule R and an instance t, we define following notations for further use:

$\mathtt{Inst}_R = \{t | R \text{ covers } t\}$ $\mathtt{Rules}_t = \{R | t \text{ is covered by } R\}$
$\mathtt{Inst}_R^+ = \{t | R \text{ correctly covers } t\}$ $\mathtt{Rules}_t^+ = \{R | t \text{ is correctly covered by } R\}$
$\mathtt{Inst}_R^- = \{t | R \text{ incorrectly covers } t\}$ $\mathtt{Rules}_t^- = \{R | t \text{ is incorrectly covered by } R\}$

4.2 Simple Precision Scoring

We now present our first candidate scoring function, which we call *simple precision scoring*. A natural score for a rule R is its *precision*: the fraction of instances covered correctly by R among the instances covered by it.

$$\mathrm{Score}_{\mathrm{SP}}(R) = \frac{|\mathtt{Inst}_R^+|}{|\mathtt{Inst}_R|} = \frac{|\mathtt{Inst}_R^+|}{|\mathtt{Inst}_R^+| + |\mathtt{Inst}_R^-|}$$

4.3 Weighted Precision Scoring

Under $\mathrm{Score}_{\mathrm{SP}}$, the score of a rule R is determined only by the number of instances covered correctly ($|\mathtt{Inst}_R^+|$) and incorrectly ($|\mathtt{Inst}_R^-|$). The nature of instances are not taken into account. The variants of $\mathrm{Score}_{\mathrm{SP}}$ proposed here assigns weights to instances, based on which the scores are computed. We assign weights to the instances based on how easy it is to

make them happy. For an instance t, define the *happiness quotient* $h(t)$ to be the fraction of rules that correctly cover t among all the rules that cover t:

$$h(t) = \frac{|\texttt{Rules}_t^+|}{|\texttt{Rules}_t|}.$$

The value $h(t)$ is a measure of how easy it is to make t happy; the larger the value of $h(t)$, it is easier to make t happy. For instance, if $h(t) \approx 1$, then $|\texttt{Rules}_t^+| \approx |\texttt{Rules}_t|$, meaning that almost any rule that covers t will annotate it correctly. Thus, it is easy to make t happy. On the other extreme, if $h(t) \approx 0$, then only a small fraction of the rules that cover t annotate it correctly. Thus it is harder to make t happy.

When we schedule a rule R, the instances in $\texttt{Inst}_R^+$ become happy and those in $\texttt{Inst}_R^-$ become unhappy. Our new scoring functions give credit to R for each instance in $\texttt{Inst}_R^+$ and award a penalty R for each instance in $\texttt{Inst}_R^-$. The credit and the penalty depend on the happiness quotient of the instance. Informally, we want to give more credit R for making hard instances happy; similarly, we want to penalize R for making easy instances unhappy. A natural way of accomplishing the above is to award a credit of $(1 - h(t))$ for each instance $t \in \texttt{Inst}_R^+$ and a penalty of $h(t)$ for each instance $t \in \texttt{Inst}_R^-$. Below, we formally define the above quantities as *gain* and *loss* associated with R. For each rule R, define

$$\begin{aligned}\mathrm{Gain}(R) &= \sum_{t \in \texttt{Inst}_R^+} (1 - h(t)) \\ \mathrm{Loss}(R) &= \sum_{t \in \texttt{Inst}_R^-} h(t)\end{aligned}$$

Based on the above quantities, we define a natural scoring function, called *Weighted Precision*:

$$\mathrm{Score}_{\mathrm{WP}}(R) = \frac{\mathrm{Gain}(R)}{\mathrm{Gain}(R) + \mathrm{Loss}(R)}$$

4.4 Refined Weighted Precision Scoring

Our third scoring function is a refinement of the weighted precision scoring. In $\mathrm{Score}_{\mathrm{WP}}$, we compute the happiness quotient of a token by taking in account the number of rules that cover the token and among those the ones that cover it correctly. The refinement is obtained by also considering the nature of these rules. We define

$$h_{\mathrm{RP}}(t) = \frac{\sum_{R \in \texttt{Rules}_t^+} \mathrm{precision}(R)}{\sum_{R \in \texttt{Rules}_t} \mathrm{precision}(R)}.$$

Gain, loss and the scoring function are defined similar to that of $\mathrm{Score}_{\mathrm{WP}}$:

$$\begin{aligned}\mathrm{Gain}_{\mathrm{RP}}(R) &= \sum_{t \in \texttt{Inst}_R^+} (1 - h_{\mathrm{RP}}(t)) \\ \mathrm{Loss}_{\mathrm{RP}}(R) &= \sum_{t \in \texttt{Inst}_R^-} h_{\mathrm{RP}}(t) \\ \mathrm{Score}_{\mathrm{RP}}(R) &= \frac{\mathrm{Gain}_{\mathrm{RP}}(R)}{\mathrm{Gain}_{\mathrm{RP}}(R) + \mathrm{Loss}_{\mathrm{RP}}(R)}\end{aligned}$$

5 Experiments

In this section, we describe rule-ordering experiments on two real-world tasks. 1) named-entity (NE) annotation that relied on hand-crafted rules for MUC-7 dataset. 2) The second application we consider is rule-based multi-class text classification. We order rules learnt on benchmark text classification datasets and observe consistent improvements by merely re-ordering rules learnt by other rule learners.

5.1 Named Entity Annotation

Rule-based named entity annotation is a natural instance of a decision list problem. Typically, rule-based NE annotation systems (Cunningham et al., 2002) require rules to be manually written as well as ordered manually. In this section, we show that our proposed rule-ordering algorithms perform better than the natural heuristic. Note that we do not intend to build a rule-based decision list which performs better than existing methods.

Setup: In our problem formulation of MAXDL , the set of instances $\mathcal{T}$ and mapping A from instances to actual annotations, together form a training set. We have access to a set of documents $D = \{d_1, d_2, \ldots, d_m\}$, that have all its named entities annotated. To generate pairs $(\mathcal{T}, A)$ using the set of documents D, let T_{d_i} represent the set of token sequences that are annotated in a document $d_i \in D$. Let $A(t)$ be the actual annotation for an instance $t \in T_{d_i}$. Given a set of rules $\mathcal{R}$ and a document collection D, each rule $R \in \mathcal{R}$ is applied to each document $d_i \in D$. The set of token sequences (instances here) which R covers ($\texttt{Inst}_R$), is included in the set of instances $\mathcal{T}$. For all instances $t \in T_{d_i}$, we add a mapping $t \rightarrow A(t)$ in A. For all other instances $t \in \{\texttt{Inst}_R - T_{d_i}\}$, we have a mapping $t \rightarrow null$ included in A. We perform these additions for each document and rule pair. Finally, we add a rule $R_* = (*, null)$ to the rule set $\mathcal{R}$. The pattern $*$ matches every instance $t \in \bigcup_{R \in \mathcal{R}, R \neq R_*} \texttt{Inst}_R$

and associates a $null$ annotation with the instance.

We only consider "person name", "organization" and "place name" annotations. We use two different rule sets containing about 30 rules each.

Table 1 presents accuracy achieved by our proposed algorithms for the two chosen rule sets. In all the cases our proposed methods perform better than $\mathrm{Score}_{\mathrm{SP}}$. The result also shows that our proposed methods generalize better than simple $\mathrm{Score}_{\mathrm{SP}}$.

Rule-sets	Accuracy	$\mathrm{Score}_{\mathrm{SP}}$	$\mathrm{Score}_{\mathrm{WP}}$	$\mathrm{Score}^{PR}_{\mathrm{WP}}$
Rule-set 1	Trng	76.4	76.7	**78.9**
	Test	50.0	52.7	**54.5**
Rule-set 2	Training	70.1	71.6	**73.3**
	Test	49.1	51.4	**52.0**

Table 1: Accuracies (in %) for different algorithms

Dataset (avg. # rules)	Acc- -uracy	JRip	$\mathrm{Score}_{\mathrm{SP}}$	$\mathrm{Score}_{\mathrm{WP}}$	$\mathrm{Score}^{PR}_{\mathrm{WP}}$
la2s (37)	Trng	86.16±0.39	86.02±0.16	86.68±0.16	**87.04±0.17**
	Test	76.93±0.43	77.88±0.16	78.05±0.17	**78.1±0.15**
oh5 (28)	Trng	86.95±0.41	88.26±0.21	88.8±0.16	**89.06±0.17**
	Test	76.43±0.58	79.08±0.37	79.37±0.38	**79.24±0.35**
tr45 (17)	Trng	91.88±0.38	92.61±0.18	92.84±0.23	**93.3±0.21**
	Test	78.9±0.47	80.99±0.29	81.19±0.28	**81.3±0.3**

Table 2: Accuracies (in %) for *RipRules*

Data set	Accu- -racy	Multi-class J48	Multi-class NaiveBayes	$\mathrm{Score}_{\mathrm{SP}}$	$\mathrm{Score}_{\mathrm{WP}}$	$\mathrm{Score}^{PR}_{\mathrm{WP}}$
la2s (18)	Trng	94.75±0.39	85.78±0.29	94.64±0.14	95.9±0.03	**95.99±0.01**
	Test	73.43±0.64	73.68±0.37	78.0±0.21	78.46±0.23	**78.64±0.29**
oh5 (30)	Trng	95.08±0.21	**99.56±0.09**	96.27±0.14	98.43±0.09	98.45±0.09
	Test	78.08±0.76	74.16±0.77	82.72±0.25	83.16±0.24	**83.98±0.26**
tr45 (30)	Trng	97.91±0.11	87.16±1.18	97.71±0.14	98.93±0.06	**98.98±0.05**
	Test	85.25±1.02	69.91±1.33	84.06±0.44	86.1±0.39	**86.42±0.41**

Table 3: Accuracies (in %) for *BinRules*

5.2 Ordering classification rules

In this section, we show another application of our algorithms in ordering classification rules. The antecedent of a classification rule is a series of tests on the input and the consequent gives the class label. Since different rules can assign conflicting classes, rule-ordering becomes important in choosing a correct class. These rules come from a variety of sources and could be hand-crafted or machine-learnt. Machine learnt rules could be generated using association mining (Agrawal and Srikant, 1994), inductive logic programming (Lavrac and Dzeroski, 1994), or Ripper (Cohen, 1995). Even classifiers can be seen as rules, *e.g.*, linear discriminants are rules that assign one of two classes to exclusive partitions of input space. Due to domain specificity and unavailability of hand-tuned rules we illustrate rule-ordering on: (1) rules induced by Ripper (Cohen, 1995) (*RipRules*), and (2) a heterogeneous set of rules obtained from naive Bayes and decision trees (*BinRules*).

Setup: We used benchmark text classification datasets (Forman, 2003) available from the Weka site[1]. These multi-class datasets represent 229 binary text classification problems, with positive class size avg. 149, and class skews avg. $1 : 31$. These are subsets of various benchmark tasks like Reuters, TREC, and Ohsumed (oh). We present only a subset of the results (with only $\mathrm{Score}_{\mathrm{WP}}$ and $\mathrm{Score}^{PR}_{\mathrm{WP}}$) here for lack of space. We report experiments over 10 random $50 : 50$ train-test splits. The training split is used to learn rules and their ordering. The orderings are evaluated on the test split and average train and test accuracies reported.

Results:

The *RipRules* setting: We induce rules (from the train split) using the JRip implementation in Weka[2] (Witten and Frank, 2005). We apply our various algorithms to merely re-order the rules output by JRip. In Table 2 we present results comparing JRip output with their re-ordered versions obtained from $\mathrm{Score}_{\mathrm{SP}}$, $\mathrm{Score}_{\mathrm{WP}}$ and $\mathrm{Score}^{PR}_{\mathrm{WP}}$. Along with the name of each data set, the average number of rules induced from the training splits are also mentioned in parentheses. The best accuracies are marked in bold. We observe that the re-ordered rule-sets using $\mathrm{Score}_{\mathrm{WP}}$ and $\mathrm{Score}^{PR}_{\mathrm{WP}}$ perform better than both baselines $\mathrm{Score}_{\mathrm{SP}}$ and JRip with lower deviations.

The *BinRules* setting: For an n-class problem we obtain classification rules by training a heterogeneous collection of one-vs-rest binary classifiers. Each classifier is either a naive Bayes or a decision tree classifier trained to discriminate one class from the rest ($2n$ classifiers). We treat each binary classifier as a classification rule that *covers* an instance if the binary classifier assigns its associated class to that instance. In addition, corresponding to every class, we introduce a default classification rule that assigns the associated class to any instance it en-

[1] http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html

[2] http://www.cs.waikato.ac.nz/ml/weka/

counters. This gives us $3n$ rules. We used the naive Bayes and J48 implementations in Weka to obtain binary rules, ordered using $\mathrm{Score}_{\mathrm{WP}}$ and $\mathrm{Score}^{PR}_{\mathrm{WP}}$, and compared with $\mathrm{Score}_{\mathrm{SP}}$ baseline in Table 3. We also show individual classifier accuracy, and the best are marked bold. It is encouraging to note that all our rule-ordering techniques always outperform their multi-class counterparts on the test data set. We outperform the baseline $\mathrm{Score}_{\mathrm{SP}}$ method on all data sets with lower deviations.

6 Conclusions

In this paper, we formulated and studied the MAXDL problem. We proved the hardness of the problem. We then proposed some heuristic approaches and established the usefulness of our methods experimentally. We observed improved performance in classification task by merely reordering the rules obtained by an existing decision list learning algorithm. In future work, we would like to explore how rule-ordering formulation can be applied to ordering heterogeneous classifiers in the ensemble learning setting.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *VLDB*, pages 487–499.

D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, D. Martin, K. Myers, and M. Tyson. 1995. Sri international fastus system: Muc-6 test results and analysis. In *MUC6 '95: Proc. of the 6th conf. on Message understanding*.

A. Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

Eric Brill. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of ANLP*.

M. E. Califf and R. J. Mooney. 1998. Relational learning of pattern-match rules for information extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*.

William W. Cohen. 1995. Fast effective rule induction. In *ICML*, pages 115–123.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of ACL*.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *JMLR Special Issue on Variable and Feature Selection*, 3:1289–1305.

M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability*. Freeman.

R. Grishman. 1997. Information extraction: Techniques and challenges. In *SCIE '97: Intnl. summer School on Information Extraction*.

Hui Han, Eren Manavoglu, C. Lee Giles, and Hongyuan Zha. 2003. Rule-based word clustering for text classification. In *SIGIR*, pages 445–446. ACM Press.

Furnkranz J. and Widmer G. 1994. Incremental reduced error pruning. In *Machine Learning: Proc. of the Eleventh International Conference*.

Nada Lavrac and Saso Dzeroski. 1994. *Inductive Logic Programming:Techniques and Applications*. Ellis Horwood, New York.

David D. Lewis, Rayid Ghani, Dunja Mladenic, Isabelle Moulinier, and Mark Wasson. 2003. Workshop on operational text classification. In *conjunction with SIGKDD*.

Hang Li and Kenji Yamanishi. 1999. Text classification using ESC-based stochastic decision lists. In *CIKM*.

D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. 2001. Named entity recognition from diverse text types. In *RANLP*.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *ANLP/NAACL 2000 Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems*.

Ronald L. Rivest. 1987. Learning decision lists. *Machine Learning*, 2(3):229–246.

Minoru Sasaki and Kenji Kita. 1998. Rule-based text categorization using hierarchical categories. In *Proceedings of SMC-98, IEEE International Conference on Systems, Man, and Cybernetics*, pages 2827–2830.

Sundar Varadarajan, Kas Kasravi, and Ronen Feldman. 2002. Text-mining: Application development challenges. In *Proceedings of the Twenty-second SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

D. Zuckerman. 2006. Linear degree extractors and the inapproximability of max-clique and chromatic number. In *STOC*.

A Re-examination of Dependency Path Kernels for Relation Extraction

Mengqiu Wang
Computer Science Department
Stanford University
mengqiu@cs.stanford.edu

Abstract

Extracting semantic relations between entities from natural language text is an important step towards automatic knowledge extraction from large text collections and the Web. The state-of-the-art approach to relation extraction employs Support Vector Machines (SVM) and kernel methods for classification. Despite the diversity of kernels and the near exhaustive trial-and-error on kernel combination, there lacks a clear understanding of how these kernels relate to each other and why some are superior than others. In this paper, we provide an analysis of the relative strength and weakness of several kernels through systematic experimentation. We show that relation extraction can benefit from increasing the feature space through convolution kernel and introducing bias towards more syntactically meaningful feature space. Based on our analysis, we propose a new convolution dependency path kernel that combines the above two benefits. Our experimental results on the standard ACE 2003 datasets demonstrate that our new kernel gives consistent and significantly better performance than baseline methods, obtaining very competitive results to the state-of-the-art performance.

1 Introduction

There exists a large body of knowledge embedded in unstructured natural language text on the Web. The sheer volume and heterogeneity of such knowledge renders traditional rule-based and manually-crafted knowledge extraction systems unsuitable. Thus it calls for methods that automatically extract knowledge from natural language text. An important step towards automatic knowledge discovery is to extract semantic relations between entities.

Two types of collections are commonly studied for relation extraction. The first type is annotated newswire text made available by programs such as Message Understanding Conferences (MUC) and Automatic Content Extraction (ACE). The types of entities that are of interest to these programs include *person*, *organization*, *facilities*, *location* and *GPE (Geo-political entities)*. Given entities in a document, the relation extraction task is to identify explicit semantic relationship such as *Located-In* and *Citizen-Of* between pairs of entities. For example, in the sentence "The funeral was scheduled for Thursday in Paris at the Saint-Germain-des-Pres Church", the organization *Saint-Germain-des-Pres Church* is "*Located-In*" GPE *Paris*. The second type of collection that has been widely studied is biomedical literature (Bunescu and Mooney, 2005b; Giuliano et al., 2006; McDonald et al., 2005b), promoted by evaluation programs such as BioCreAtIvE and JNLPBA 2004. In this particular domain, studies often focus on specific entities such as genes and proteins. And the kinds of relations to extract are usually gene-to-protein interactions.

The predominant approach to relation extraction treats the task as a multi-class classification problem, in which different relation types form different output classes. Early work employed a diverse range of features in a linear classifier (commonly referred to as "feature-based" approaches), including lexical features, syntactic parse features, dependency features and semantic features (Jiang and Zhai, 2007; Kambhatla, 2004; Zhou et al., 2005). These approaches were hindered by drawbacks such as limited feature space and excessive feature engineering. Kernel methods (Cortes and Vapnik, 1995; Cristianini and Shawe-Taylor, 2000) on the other hand can explore a much larger feature space very efficiently. Recent studies on relation extraction have shown that by combining kernels with Support-vector Machines (SVM), one can obtain results superior to feature-based methods (Bunescu

and Mooney, 2005b; Bunescu and Mooney, 2005a; Culotta and Sorensen, 2004; Cumby and Roth, 2003; Zelenko et al., 2003; Zhang et al., 2006a; Zhang et al., 2006b; Zhao and Grishman, 2005).

Despite the large number of recently proposed kernels and their reported success, there lacks a clear understanding of their relative strength and weakness. In this study, we provide a systematic comparison and analysis of three such kernels — subsequence kernel (Bunescu and Mooney, 2005b), dependency tree kernel (Culotta and Sorensen, 2004) and dependency path kernel (Bunescu and Mooney, 2005a). We replicated these kernels and conducted experiments on the standard ACE 2003 newswire text evaluation set. We show that whereas some kernels are less effective than others, they exhibit properties that are complementary to each other. In particular, We found that relation extraction can benefit from increasing the feature space through convolution kernel and introducing bias towards more syntactically meaningful feature space.

Drawn from our analysis, we further propose a new convolution dependency path kernel which combines the benefits of the subsequence kernel and shortest path dependency kernel. Comparing to the previous kernels, our new kernel gives consistent and significantly better performance than all three previous kernels that we look at.

2 Related Work

Statistical methods for relation extraction can be roughly categorized into two categories: feature-based and kernel-based.

Feature-based methods (Jiang and Zhai, 2007; Kambhatla, 2004; Zhou et al., 2005) use pre-defined feature sets to extract features to train classification models. Zhou et al. (2005) manually crafted a wide range of features drawn from sources such as lexical, syntactic and semantic analyses. Combined with SVM, they reported the best results at the time on ACE corpus. Kambhatla (2004) took a similar approach but used multivariate logistic regression (Kambhatla, 2004). Jiang & Zhai (2007) gave a systematic examination of the efficacy of unigram, bigram and trigram features drawn from different representations — surface text, constituency parse tree and dependency parse tree.

One drawback of these feature-based methods is that the feature space that can be explored is often limited. On the other hand, kernel-based methods offer efficient solutions that allow us to explore a much larger (often exponential, or in some cases, infinite) feature space in polynomial time, without the need to explicitly represent the features.

Lodhi et al. (2002) described a convolution string kernel, which measures the similarity between two strings by recursively computing matching of all possible subsequences of the two strings. Bunescu & Mooney (2005b) generalized the string kernel to work with vectors of objects occurred in relation extraction. In a later work also done by Bunescu & Mooney (2005a), they proposed a kernel that computes similarities between nodes on the shortest dependency paths that connect the entities. Their kernel assigns no-match to paths that are of different length. And for paths that are of the same length, it simply computes the product of the similarity score of node pairs at each index. The dependency tree kernel proposed by Zelenko et al. (2003) was also inspired by the string kernel of Lodhi et al. (2002). Their kernel walks down the parse trees from the root and computes a similarity score for children nodes at each depth level using the same subsequence algorithm as the string kernel. Culotta & Sorensen (2004) worked on the same idea but applied it to dependency parse trees. Prior to these two tree kernels, Collins & Duffy (2001) proposed a convolution tree kernel for natural language tasks. Their kernel has since been applied to relation extraction by Zhang et al. (2006a). The tree kernel considers matching of all subtrees that share the same production rule at the root of the subtree. Zhang et al. (2006a) showed results that are significantly better than the previous two dependency tree kernels. They obtained further improvements in their later paper (2006b) by composing the tree kernel with a simple entity kernel and raising the composite kernel to polynomial degree 2. Another study on kernel composition is the work by Zhao & Grishman (2005).

It is worth noting that although there exist standard evaluation datasets such as ACE 2003 and 2004, many of the aforementioned work report results on non-standard datasets or splits, making it difficult to directly compare the performance. We

feel that there is a sense of increasing confusion down this line of research. Although partly due to the lack of compatibility in evaluation results, we believe it is more due to the lack of understanding in the relative strength and weakness of these kernels. Therefore we focus on analyzing and understanding the pros and cons of different kernels, through systematic comparison and experimentation.

3 Kernel Methods for Relation Extraction

In this Section we first give a very brief introduction to kernel methods. We then present the algorithms behind three kernels that we are particularly interested in: subsequence kernel (Bunescu and Mooney, 2005b), dependency tree kernel (Culotta and Sorensen, 2004) and shortest path dependency kernel (Bunescu and Mooney, 2005a).

3.1 SVM and Kernels

Support-Vector Machines (Cortes and Vapnik, 1995; Cristianini and Shawe-Taylor, 2000) learn to find hyperplanes that separate the positive and negative data points so that the margin between the support-vector points and the hyperplane is maximized. The dual formulation of the optimization problem involves only computing the dot product of feature vectors. This is equivalent to mapping the data points into a high dimensional space. And the separating plane learnt in the high dimensional space can give non-linear decision boundaries. The dot product of data points can be computed using a kernel function $K(X,Y) = \langle \phi(X), \phi(Y) \rangle$ for any mapping function. A valid kernel function satisfies certain properties: it is symmetric and the *Gram* matrix G formed by $K(X,Y)$ is positive semi-definite.

3.2 Subsequence Kernel

The subsequence kernel introduced in (Bunescu and Mooney, 2005b) is a generalization of the string kernel first introduced by Lodhi et al. (2002). The feature space of the original string kernel Σ_{string_kernel} is defined as $\Sigma_{string_kernel} = \Sigma_{char}$, where Σ_{char} is simply a set of characters. Bunescu & Mooney (2005a) re-defined the feature space to be $\Sigma_x = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_k$, where $\Sigma_1, \Sigma_2, \cdots, \Sigma_k$ can be some arbitray disjoint feature spaces, such as the set of words, part-of-speech (POS) tags, etc. We can measure the number of common features shared by two feature vectors $x, y \in \Sigma_x$ using function $c(x,y)$. Let s,t be two sequences over the feature set Σ_x, we use $|s|$ to denote the length of s. Thus s can be written out as $s_1 \cdots s_{|s|}$. We use $s[i:j]$ to denote a continuous subsequence $s_i \cdots s_j$ of s. Let $\mathbf{i} = (i_1, \cdots, i_{|\mathbf{i}|})$ be a sequence of $|\mathbf{i}|$ indices in s, we define the *length* of the index sequence $\mathbf{i}$ to be $l(\mathbf{i}) = i_{|\mathbf{i}|} - i_1 + 1$. Similarly we have index sequence $\mathbf{j}$ in t of length $l(\mathbf{j})$.

Let $\Sigma_\cup = \Sigma_1 \cup \Sigma_2 \cup \cdots \cup \Sigma_k$ be the set of all possible features. A sequence $u \in \Sigma_\cup^*$ is a subsequence of feature vector sequence s if there exists a sequence of $|u|$ indices $\mathbf{i}$, such that $u_k \in s_{i_k}$, $\forall k \in \{1, \cdots, |u|\}$. Follow the notions in (Bunescu and Mooney, 2005b; Cumby and Roth, 2003), we use $u \prec s[\mathbf{i}]$ as a shorthand for the above component-wise '$\in$' relationship. Now we can define the kernel function $K_n(s,t)$ to be the total number of weighted common subsequence of length n between the two sequeneces s and t.

$$K_n(s,t) = \sum_{u \in \Sigma_\cup^n} \sum_{\mathbf{i}: u \prec s[\mathbf{i}]} \sum_{\mathbf{j}: u \prec t[\mathbf{j}]} \lambda^{l(\mathbf{i}) + l(\mathbf{j})} \quad (1)$$

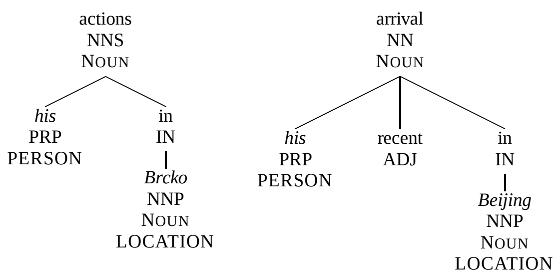
where λ is a decaying factor ≤ 1, penalizing long, sparse subsequence. We can re-write this kernel function as

$$K_n(s,t) = \sum_{\mathbf{i}:|\mathbf{i}|=n} \sum_{\mathbf{j}:|\mathbf{j}|=n} \prod_{k=1}^{n} c(s_{i_k}, t_{j_k}) \lambda^{l(\mathbf{i}) + l(\mathbf{j})} \quad (2)$$

(Bunescu and Mooney, 2005b) showed that using the recursive dynamic programming algorithm from (Cumby and Roth, 2003), the kernel $K_n(s,t)$ can be computed in $O(kn|s||t|)$ time.

3.3 From Subsequence to Tree Kernels

We will use an example to illustrate the relation between the dependency tree kernels proposed by (Culotta and Sorensen, 2004; Zelenko et al., 2003) and the subsequence kernel we introduced above. Consider two instances of the "Located-In" relations "*his* actions in *Brcko*" and "*his* recent arrival in *Beijing*". The dependency parse trees of these two sentences are shown below.



The entities in these two relations are the pronoun mentions of "*his*", and two locations "*Brcko*" and "*Beijing*", all shown in italic. The dependency tree kernel visits nodes in the two trees starting from the root. And at each depth level, it takes nodes that are at that level and form two sequences of nodes. For example, in the example instances, nodes at one level below the root forms vectors s=⟨{*his*, PRP, PERSON},{in, IN}⟩ and t=⟨{*his*,PRP,PERSON},{recent, ADJ},{in, IN}⟩. It then makes use of the subsequence kernel in the previous section to compute the total number of weighted subsequences between these two vectors. The kernel returns the sum of subsequence matching scores at each depth level as the final score.

3.4 Shortest Path Dependency Kernel

The shortest path dependency kernel proposed by Bunescu & Mooney (2005a) also works with dependency parse trees. Reuse our example in the previous section, the shortest dependency path between entity *his* and *Brcko* in the first sentence is s=⟨{*his*, PRP, PERSON}, {actions, NNS, NOUN}, {in, IN}, {*Brcko*, NNP, NOUN, LOCATION}⟩; and the path between *his* and *Beijing* in the second sentence is t=⟨{*his*, PRP, PERSON}, {arrival, NN, NOUN}, {in, IN}, {*Beijing*, NNP, NOUN, LOCATION}⟩. Since most dependency parser output connected *trees*, finding the shortest path between two nodes is trivial. Once the two paths are found, the kernel simply computes the product of the number of common features between a pair of nodes at each index along the path. If the two paths have different number of nodes, the kernel assigns 0 (no-match) to the pair. Formally, the kernel is defined as:

$$K(s,t) = \begin{cases} 0, & \text{if } |s| \neq |t| \\ \prod_{i=1}^{n} c(s_i, t_i), & \text{if } |s| = |t| \end{cases} \qquad (3)$$

kernel method	5-fold CV on ACE 2003		
	Precision	Recall	F1
subsequence	0.703	**0.389**	0.546
dependency tree	0.681	0.290	0.485
shortest path	**0.747**	0.376	**0.562**

Table 1: Results of different kernels on ACE 2003 training set using 5-fold cross-validation.

4 Experiments and Analysis

We implemented the above three kernels and conducted a set of experiments to compare these kernels. By minimizing divergence in our experiment setup and implementation for these kernels, we hope to reveal intrinsic properties of different kernels.

4.1 Experiment setup

We conducted experiments using the ACE 2003 standard evaluation set. Training set of this collection contains 674 doc and 9683 relations. The test set contains 97 doc and 1386 relations. 5 entity types (Person, Organization, Location, Facilities and Geopolitical Entities) and 5 top-level relation types (At, Near, Part-of, Role and Social) are manually annotated in this collection. Since no development set is given, we report results in this section only on the training set, using 5-fold cross-validation, and defer the comparison of results on the test set till Section 6. Corpus preprocessing is done as the following: sentence segmentation was performed using the tool from CCG group at UIUC [1]; words are then tokenized and tagged with part-of-speech using MXPOST (Ratnaparkhi, 1996) and dependency parsing is performed using MSTParser (McDonald et al., 2005a). We used the SVM-light (Joachims, 2002) toolkit and augmented it with our custom kernels. SVM parameters are chosen using cross-validation (C=2.4), and the decaying factor in all kernels are uniformally set to be 0.75. We report precision (P), recall (R) and F-measure (F) on the training (5-fold cross-validation) and test set.

4.2 Comparison of Kernels

In table 1 we listed results of the above three kernels on the training set using 5-fold cross-validation. A

[1] `http://l2r.cs.uiuc.edu/~cogcomp/atool.php?tkey=SS`

first glimpse of the results tells us that the shortest path kernel performs the best in terms of F-measure, while the dependency tree kernel did the worst. The performance of subsequence kernel is not as good as the dependency path kernel, but the difference is small. In particular, the subsequence kernel gave the best recall, whereas the dependency path kernel gave the highest precision.

To understand why shortest path kernel performs better than the subsequence kernel, let us review the definition of these two kernels. The subsequence kernel considers all subsequences of feature vector sequences that are formed by all words occurred in-between two entities in a sentence; while the shortest path kernel only considers feature vector sequences formed by words that are connected through a dependency path. In general, the sequences considered in the dependency path kernel are more compact than the sequences used in the subsequence kernel. Actually, in most cases the dependency path sequence is indeed *one particular subsequence* of the entire subsequence used in subsequence kernel. Arguably, this particular subsequence is the one that captures the most important syntactic information. Although the feature spaces of the dependency path kernels are not subsets of the subsequence kernel, we can clearly see that we get higher precisions by introducing bias towards the syntactically more meaningful feature space.

However, the dependency path kernel is fairly rigid and imposes many hard constraints such as requiring the two paths to have exactly the same number of nodes. This restriction is counter-intuitive. To illustrate this, let us reconsider the example given in Section 3. In that example, it is obviously the case that the two instances of relations have very similar dependency path connecting the entities. However, the second path is one node longer than the first path, and therefore the dependency path kernel will declare no match for them. The subsequence kernel, on the other hand, considers subsequence matching and therefore inherently incorporates a notion of fuzzy matching. Furthermore, we have observed from the training data that many short word sequences carry strong relational information; hence only part of the entire dependency path is truly meaningful in most cases. It also helps to understand why subsequence kernel has better recall than dependency path kernel.

	ACE 2003 test set		
kernel method	Precision	Recall	F1
subsequence	0.673	0.499	0.586
dependency tree	0.621	0.362	0.492
shortest path	0.691	0.462	0.577
convolution dep. path	**0.725**	**0.541**	**0.633**
(Zhang et al., 2006b)	*0.773*	*0.656*	*0.709*

Table 2: Results on the ACE 2003 test set. We reference the best-reported score (in italic) on this test set, given by (Zhang et al., 2006b)

The disappointing performance of the dependency tree kernel can also be explained by our analysis. Although the dependency tree kernel performs subsequence matching for nodes at each depth level, it is unclear what the relative syntactic or semantic relation is among sibling nodes in the dependency tree. The sequence formed by sibling nodes is far less intuitive from a linguistic point of view than the sequence formed by nodes on a dependency path.

To summarize the above results, we found that dependency path kernel benefits from a reduction in feature space by using syntactic dependency information. But the subsequence kernel has an edge in recall by allowing fuzzy matching and expanding the feature space into convolution space. We will show in the following section that these two benefits are complementary and can be combined to give better performance.

5 Combining the Benefits – A New Kernel

It is a natural extension to combine the two benefits that we have identified in the previous section. The idea is simple: we want to allow subsequence matching in order to gain more flexibility and therefore higher recall, but constrain the sequence from which to deduce subsequences to be the dependency path sequence. We call the combined kernel a "convolution dependency path kernel".

6 Final Test Results

We obtained the final results on the test set of the ACE 2003 collection, using the same experimental setting as above. The results are listed in Table 2. From the table we can see that the performances of the previous three kernels hold up qualitatively on

the test set as cross-validation on training set. There is one exception that the shortest path kernel's F-measure score is no longer better than the subsequence kernel on the test set, but the difference is small. And our new convolution dependency path kernel beats all above three kernels in precision, recall and F-measure, suggesting that our analysis is accurate and the benefits we outlined are truly complementary.

Comparing to the best reported results on the same test set from (Zhang et al., 2006b), our scores are not as high, but the results are quite competitive, given our minimum efforts on tuning kernel parameters and trying out kernel combinations.

7 Conclusion

We re-examined three existing kernel methods for relation extraction. We conducted experiments on the standard ACE 2003 evaluation set and showed that whereas some kernels are less effective than others, they exhibit properties that are complementary to each other. In particular, we found that relation extraction can benefit from increasing the feature space through convolution kernel and introducing bias towards more syntactically meaningful feature space. Drawn from our analysis, we proposed a new convolution dependency path kernel which combines the benefits of the subsequence kernel and shortest path dependency kernel. Comparing with previous kernels, our new kernel consistently and significantly outperforms all three previous kernels, suggesting that our analyses of the previously proposed kernels are correct.

References

R. C. Bunescu and R. J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT/EMNLP*.

R. C. Bunescu and R. J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings of NIPS*.

M. Collins and N. Duffy. 2001. Convolution kernels for natural language. In *Proceedings of NIPS*.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

N. Cristianini and J. Shawe-Taylor. 2000. *An Introduction to Support-vector Machines*. Cambridge University Press.

A. Culotta and J. Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*.

C. M. Cumby and D. Roth. 2003. On kernel methods for relation learning. In *Proceedings of ICML*.

C. Giuliano, A. Lavelli, and L. Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *Proceedings of EACL*.

J. Jiang and C. Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of NAACL-HLT*.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines*. Ph.D. thesis, Universit ̈at Dortmund.

N. Kambhatla. 2004. Combining lexical, syntactic and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL*.

H. Lodhi, C Saunders, J. Shawe-Taylor, N. Cristianini, and C Watkins. 2002. Text classification using string kernels. *JMLR*, 2:419–444.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

R. McDonald, F. Pereira, S. Kulick, S. Winters, Y. Jin, and P. White. 2005b. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of ACL*.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *JMLR*, 3:1083–1106.

M. Zhang, J. Zhang, and J. Su. 2006a. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of NAACL-HLT*.

M. Zhang, J. Zhang, J. Su, and G. Zhou. 2006b. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of ACL*.

S. Zhao and R. Grishman. 2005. Extraction relations with integrated information using kernel methods. In *Proceedings of ACL*.

G. Zhou, S. Jian, J. Zhang, and M. Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL*.

Mining Chinese-English Parallel Corpora from the Web

Bo Li
School of Computer Science
Wuhan University
Wuhan, 430072, China
whulibo@gmail.com

Juan Liu
School of Computer Science
Wuhan University
Wuhan, 430072, China
liujuan@whu.edu.cn

Abstract

Parallel corpora are a crucial resource in research fields such as cross-lingual information retrieval and statistical machine translation, but only a few parallel corpora with high quality are publicly available nowadays. In this paper, we try to solve the problem by developing a system that can automatically mine high quality parallel corpora from the World Wide Web. The system contains a three-step process. The system uses a web spider to crawl certain hosts at first. Then candidate parallel web page pairs are prepared from the downloaded page set. At last, each candidate pair is examined based on multiple standards. We develop novel strategies for the implementation of the system, which are then proved to be rather effective by the experiments towards a multilingual website.

1 Introduction

Parallel corpora consisting of text in parallel translation plays an important role in data-driven natural language processing technologies such as statistical machine translation (Brown et al., 1990) and cross-lingual information retrieval (Landauer and Littman, 1990; Oard, 1997). But the fact is that only a few parallel corpora with high quality are publicly available such as the United Nations proceedings and the Canadian Parliament proceedings (LDC, 1999). These corpora are usually small in size, specializing in narrow areas, usually with fees and licensing restrictions, or sometimes out-of-date. For language pairs such as Chinese and English, the lack of parallel corpora is more severe. The lack of such kind of resource has been an obstacle in the development of the data-driven natural language processing technologies. But the intense human labor involved in the development of parallel corpora will still make it very hard to change the current situation by hand.

The number of websites containing web pages in parallel translation increases considerably these years, which gives hope that we can construct parallel corpora with high quality in a big scale more easily. In this paper, we present a system named Parallel Corpus Mining System (PCMS) which can automatically collect Chinese-English parallel web corpora from the Web. Similar with previous work, PCMS uses a three-step process. First, the web spider WebZip[1] is used to crawl the hosts specified by users. In the second step, candidate parallel web page pairs are prepared from the raw web page set fetched based on some outer features of the web pages. A novel strategy is designed to utilize all these features to construct high quality candidate parallel page pairs, which can raise the performance and reduce the time complexity of the system. In the third step, candidate page pairs are evaluated based on multiple standards in which page structure and content are both considered. The actually parallel page pairs are saved.

The content-based strategy in the PCMS system is implemented mainly based on the vector space model (VSM). We design a novel implementation of VSM to bilingual text, which is called bilingual vector space model (BVSM). In previous content-based work, they usually use coarse criterions to measure the similarity of bilingual text. For exam-

[1] http://www.spidersoft.com/webzip/default.asp

ple, Ma and Liberman (1999) measured the content similarity by the count of parallel token pairs in the text which are weak at representing the actual content of the text. VSM was considered for evaluating the similarity of bilingual text in (Chen et al., 2004), but unfortunately the particular description of the implementation which was a bit complex was not mentioned in their work, and the time complexity of their system was rather high. Besides, there are also some other types of methods for mining parallel corpora from the web such as the work in (Resnik, 1998), (Resnik and Smith, 2003) and (Zhang et al., 2006). Most of these methods are unbalanced between precision and recall or computationally too complex. We detail the implementation of BVSM in the PCMS system in this paper. The experiments conducted to a specific website show that PCMS can achieve a better overall result than relative work reported.

The structure of the paper is as follows. The system architecture of PCMS is introduced in Section 2. We introduce the details of the step for preparing candidate web page pairs in Section 3. The next step, candidate page pair evaluation, is described in Section 4. We discuss the results of the experiments and conclude the paper in the last two sections.

2 The PCMS System

The PCMS system is designed to mine parallel corpora automatically from the web. As has been clarified above, the system employs a three-step process. The first is a web page fetching step. There are some tools to do the job and the PCMS system uses WebZip to fetch all the web pages from specific hosts. We usually choose some sites which probably contain high quality parallel web pages such as the site of the ministry of foreign affairs of China. After the web pages are obtained from the servers, the web pages which are too small, for example smaller than 5k bytes, are excluded from the page set. Then for each page in the page set, the HTML source of the web page is parsed and the noise such as the advertisement is excluded from the raw web page. The second is the candidate parallel page pair preparation step. The web pages are paired according to the URL similarity and some other features of the web pages. The third is the candidate parallel page pair evaluation step which is the key section of the PCMS system. Both web page structure and content are considered in this step. The candidate parallel page pairs prepared by the second step are first filtered by the structure-based criterion and then evaluated by the content-based criterion. We develop novel strategies for the third step and describe it in detail in the following sections.

3 Candidate Parallel Pair Preparation

The web spider can fetch a great many web pages in different languages from certain hosts. Usually the language of a web page can be identified by some feature strings of the URL. For example, the URLs of many English web pages contain strings such as *e, en, eng* and *english* which are called *language identification strings*. The *language identification strings* are usually attached to the other part of the URL with symbols such as '–', '/' and '_'. The number of web pages downloaded by the web spider is very large, so the pairs produced will be a huge amount if we treat each web page in language *A* and each in language *B* as a candidate pair, which will then make the third step of the system computationally infeasible. Parallel web pages usually have similar URLs. For example, the web page *P1* in Chinese and *P2* in English are parallel:

Web page *P1* URL[2]:
www.fmprc.gov.cn/chn/wjdt/wshd/t358904.htm
Web page *P2* URL:
www.fmprc.gov.cn/eng/wjdt/wshd/t358905.htm

We can see that the URL of page *P1* and the URL of page *P2* share most of the strings such as *www.fmprc.gov.cn*, *wjdt,* and *wshd*. In some other cases, the similarity between the URLs of parallel web pages may be not that direct but should still be obvious.

In PCMS, a novel strategy is designed to measure the URL similarity of the candidate web page pair. Before the URL similarity evaluation process, the *language identification strings* of the URLs should be substituted by a uniform string which seldom occurs in normal URLs. For example, the *language identification strings* such as *en, eng, cn* and *chn* are substituted by the string *** which seldom occurs in normal URLs. For example, the above page *P1* after the URL substitution process is *www.fmprc.gov.cn/***/wjdt/wshd/t358904.htm.* After the substitution process, the similarity of the

[2] The protocol string HTTP is omitted here.

new URLs is evaluated. For evaluating the URL similarity of web page *PA* in language *A* and web page *PB* in language *B,* the following criterions are considered.

Criterion 1: *URL length difference.*

It can be found that the length of the URLs of parallel web pages is usually similar. The length of the URL here refers to the number of directories in the URL string. For example, the URL of the above web page *P1* contains the directories ***[3], *wjdt* and *wshd*, and then the URL length of *P1* is 3. If two web pages *PA* and *PB* are parallel, the URL length of *PA* and *PB* should be similar. The *URL length difference* criterion is define as

$$URL\ diff\,(PA,PB)=\frac{|len(PA)-len(PB)|}{len(PA)+len(PB)} \quad (1)$$

where *URL diff(PA,PB)* is the *URL length difference* between *PA* and *PB, len(PA)* is the URL length of page *PA* and *len(PB)* is the URL length of *PB.* The value of *URL length difference* is between 0 and 1, and the more similar two URLs are, the smaller the value is. If the URL lengths of *PA* and *PB* are the same, the *URL length difference* between *PA* and *PB* should be 0.

Criterion 2: *URL directory similarity.*

Besides URL length, URL directory information is also considered in the candidate page pair preparation step. It can be observed that the URLs of parallel web pages usually share similar directory structure which can be represented by the common directories in the URLs. For example, the above web page *P1* and web page *P2* share the directories ****, wjdt* and *wshd.* To measure the *URL directory similarity* of the web page *PA* and the web page *PB*, a criterion is defined as

$$URL\ dirsim(PA,PB)=\frac{2*comdir(PA,PB)}{len(PA)+len(PB)} \quad (2)$$

where *URL dirsim(PA,PB)* is the *URL directory similarity* of page *PA* and page *PB*, *comdir(PA,PB)* is the number of common directories *PA* and *PB* share, *len(PA)* and *len(PB)* are the same as above. The value of *URL directory similarity* is between 0 and 1. The bigger the value is, the more similar the two pages are. When two web pages have the same URLs, the *URL directory similarity* should be 1.

[3] The *language identification strings* of the URL have been substituted by the uniform string ***.

Criterion 3: *Similarity of some other features.*

Some other features such as the file size of the web page and the time the page created can help to filter the nonparallel web page pairs with low cost.

Based on the combination of the above criterions, the web page pairs of which the similarity excesses certain threshold are treated as the candidate parallel pairs, which are then to be processed by the following evaluation step.

4 Candidate Parallel Pair Evaluation

It is the key section of the system to evaluate the candidate parallel web page pairs. Though content-based methods are what the candidate parallel page pair evaluation step mainly relies on, the structure of the web pages is also considered in the evaluation step of the PCMS system for it can help to filter out some page pairs that are obviously non-parallel at low cost. The candidate parallel page pair set is first filtered by the structure-based strategy which is similar with the one in (Resnik, 1998), and we consider some more structure relative features such as color and font. A loose constrain is set on the structure similarity criterion, because it is merely a preliminary filter step to reduce the scale of the problem.

After the structure-based filter stage, the page pairs left are then to be evaluated by the content-based stage which is the key of the candidate parallel page pair evaluation step. The performance of the PCMS system relies mainly on this module. In the content-based stage, the candidate page pairs are first filtered based on some content related features and then the page pairs left are evaluated by the BVSM model.

4.1 The Content Related Feature-based Filter

In the first part of the content-based strategy, some content related features such as time stamp and navigation text are combined to construct a preliminary step to filter the candidate page pair set and reduce the number of pairs to be processed by BVSM. Many web pages contain time stamps which identify the time when the web pages were constructed. If two pages are parallel, the time when they are constructed should be similar. Navigation text usually demonstrates the type information of the content of the web page. For example, a web page with anchor text *Home-News-China* is probable about the news which happened in China.

So if two web pages are parallel, their navigation text if there is any should be similar. To evaluate the similarity of two pieces of navigation text in two languages, we need a bilingual navigation text wordlist. For each layer, for example *news*, in one navigation text, if its translation *新闻 xin-wen* appears in the other navigation text, the similarity *count* will be added by 1. The similarity between two pieces of navigation text is defined as

$$similarity = \frac{2 * count}{layer_{NC} + layer_{NE}} \quad (3)$$

where $layer_{NC}$ demonstrates the layer count of the navigation text of the Chinese web page and $layer_{NE}$ is that of the English web page. For example, the $layer_{NE}$ of the navigation text *Home-News-China* is 3. If the similarity gotten from formula (3) is below certain threshold, the corresponding web page pair will not be considered as parallel.

4.2 The BVSM Model

In the second part of the content-based strategy, BVSM is implemented to evaluate the similarity of candidate parallel page pairs. VSM is an important technology for representing text and has been applied to some other research areas. But this model is usually applicable to monolingual text processing problem. For bilingual text processing, we should design a new strategy to use VSM for the new problem. A bilingual dictionary is a must for importing VSM to bilingual problem. We give a brief introduction to the bilingual dictionary we use first. Each *entry line* of the dictionary consists of three parts. The first part is the English word, the middle is a list separator and the last is the corresponding Chinese word. A sample of the dictionary can be found in *Appendix A.* For each English word, there may be some Chinese words serving as its translations. The same conclusion can be gotten for each Chinese word.

Based on the bilingual dictionary, we can represent the Chinese and English web pages as vectors respectively. First, we give every English word in the bilingual dictionary a unique ID according to its position in the dictionary beginning from 1. For example, the ID of the English word in the first row is 1, and the ID of the next new English word in the dictionary is 2 and so forth. For convenience, we denote the Chinese web page as *C* and the English web page as *E* in each web page pair. We then can represent each web page as follows.

For *E,* we extract all the words from the web page and stem them first. The length of the vector of *E* equals the length of the bilingual dictionary which is the number of the different English words in the dictionary. For each dimension of the vector, for example *k,* we assign the number of the words with ID *k* occurring in all the words extracted to it. If certain words in the bilingual dictionary never occur in *E,* we assign the value 0 to the corresponding dimensions which are identified by the IDs of those words. If some words in *E* haven't occurred in the dictionary, we just ignore them.

For *C*, the procedure to construct a vector is more complex. In the PCMS system, the procedures of word segmentation and POS for Chinese are finished in a single run. The length of the vector of *C* equals to that of the vector of *E*. As has been pointed out, one Chinese word may correspond to more than one English word in the bilingual dictionary. For example in *Appendix A,* the Chinese word *放弃 fang-qi* corresponds to *abandon*, *depart* and *leave*. In the vector of *E*, each dimension strands for the count of a single English word with a unique ID occurring in the English text. In order to construct a vector for *C* which is comparable to the vector of *E*, a single Chinese word in *C* should contribute to more than one dimension of the vector of *C*. In order to distribute the count/weight of each Chinese word to the corresponding dimensions of the vector of *C,* we first count the number of each entry which is a Chinese word with a specific POS, for example *(放弃, Verb)*, in *C*. Then for each entry, we distribute its count to all the dimensions identified by the IDs of the English words which the Chinese word in the entry corresponds to. The count distribution process is detailed below.

If the Chinese word in the entry *Cent* is a content word which we call here to mean that it carries the main content of a language including noun, verb and adjective, we will divide the corresponding English words in the bilingual dictionary into four separate classes: the words that haven't appeared in the English text (C_4), the words that have the same POS with the entry (C_1), the words that have similar POS with the entry (C_2) and the other words (C_3). For convenience, the count of the entry *Cent* in *C* is denoted as N_{1234}. If the capacity of C_4 is 0 which means there are no words belonging to the class C_4, then N_{1234} is all devoted to the words

in C_1, C_2 and C_3, else a certain proportion, for example 10%, of N_{1234} is assigned to all the words in C_4 averagely and the left of N_{1234} is assigned to the words in C_1, C_2 and C_3. Similarly, we denote the count left to words in C_1, C_2 and C_3 as N_{123}, and then if the capacity of C_3 is 0, N_{123} is all denoted to the words in C_1 and C_2, else a certain proportion of N_{123} is denoted to all the words in C_3 averagely and the left of N_{123} is devoted to the words in C_1 and C_2. For words in C_1 and C_2, the count distribution strategy is similar.

If the Chinese word in the entry *Cent* is not a content word, we classify the corresponding English words into two classes: the words that haven't appeared in the English text (C_2) and the other words (C_1). The same method as above is used to distribute the count.

4.3 Similarity Evaluation Criterions

Based on the above strategies, the two web pages can be represented by their vectors respectively. Then the next step is to calculate the similarity of the two vectors, which is also the similarity of the two web pages. Some comments were given on different similarity measures such as *Euclidence distance*, *Inner product*, *Cosine coefficient*, *Dice coefficient* and *Jaccard coefficient* in (Chen et al., 2004). It was suggested that for a pair of documents to be considered parallel, we could expect that these two documents contained the two corresponding sets of translated terms and each corresponding term was carrying an identical contextual significance in each of the document respectively. For that, the *Jaccard coefficient* is more appropriate for the calculation of the similarity score. While in our experiments, we find that *Cosine coefficient* is more suitable. Because the size of the bilingual dictionary is small and we exclude all the words which are not in the dictionary from the text of the web pages when we construct the vectors, it is possible that the counterparts of some words in one web page can not be found in its corresponding web page. Though we have done some smooth work in the BVSM model, there is still a gap between the assumptions by Chen et al. (2004) and the situation of our problem. The second reason we think is that the translation process by human is almost sentence to sentence, but not word to word. As a result, it is normal that there are no words in one language serving as the translation for certain words in the other language. Based on the *Cosine coefficient* criterion, the similarity between two vectors which are represented by $(x_1, x_2, x_3, \ldots, x_p)$ and $(y_1, y_2, y_3, \ldots, y_p)$ respectively is

$$\cos ine\,coefficient = \frac{\sum_{i=1}^{p} x_i y_i}{\sqrt{\sum_{i=1}^{p} x_i^2 * \sum_{i=1}^{p} y_i^2}} \qquad (4)$$

The similarity measure is between 0 and 1, and the bigger the value is, the more similar the two vectors are. We set a certain threshold for the similarity measure based on our experience in PCMS.

5 Experiments and Discussion

In this section, we practice the experiments designed to evaluate the performance of the PCMS system and compare it with similar work earlier.

5.1 Evaluation Standards

Precision and *recall* are two widely used evaluation standards in the area of natural language processing. In our experiments, we define *precision* as the proportion of page pairs in parallel translation to the total page pairs produced by the PCMS system. *Recall* is defined as the proportion of page pairs in parallel translation produced by the PCMS system to the total parallel page pairs in the whole web page set.

The number of pairs in parallel translation should be calculated from the human annotated page pairs. We ask a native Chinese speaker who has a fluent English tongue to annotate these page pairs. To calculate the *recall*, we need to know the number of parallel pairs in the web page set. It is hard to count out the actual number of the parallel pairs in the page set because the web page set is really too big. We build a relatively smaller test set to test the *recall* of the PCMS system.

5.2 Parallel Corpus Construction

In order to construct a high quality parallel corpus in the experiments, the website of the ministry of foreign affairs of China (*http://www.fmprc.gov.cn*) is chosen to be crawled. After the rough observation, it is found that a huge number of web pages fetched are in parallel translation. We get a web page set consisting of 40262 Chinese web pages and 17324 English web pages by the tool WebZip. After the preprocess step, the web pages left are to

be examined by the core modules of PCMS. It takes nearly 3 hours to finish the task on a PC with a P4 2.0G CPU and 512MB RAM, which is faster than the early systems. To evaluate the *precision* of the system, we randomly choose a subset of the web page pairs which PCMS gives as output, and get a web page set of 500 web page pairs. We manually annotate it and find that there are 479 truly parallel page pairs among them. Then the *precision* is about 96%. We analysis the 21 non-parallel pairs the PCMS system gives and find that most of these web pages are short web pages containing limited text. To obtain the *recall* of the PCMS system, we construct a test page set consisting of 350 parallel page pairs and 150 nonparallel page pairs. The ratio 350/150 is decided based on rough estimation of the whole page set. The PCMS system is examined on the test set, which produces 337 page pairs which are truly parallel, thus a *recall* of 96%. We analysis the 13 parallel pages which are recognized as nonparallel by the PCMS system and find that most of them are short web pages. We then come to the conclusion that the drawback that BVSM is weak at representing short text leads to the system's failure to identify the parallel web page pairs. Though the model has some drawbacks, the overall result consisting of performance and time complexity is much better than the former similar work.

6 Conclusion

The paper presents a web-based parallel corpus construction system PCMS. The system first fetches all the web pages from specific hosts, and then prepares candidate parallel web page pairs based on features such as URL and web page file size. At last the candidate pairs are examined by a two-stage similarity evaluation process in which the structure and content of the web pages are both considered. To enhance the performance of the PCMS system, we design some novel strategies for the implementation of these steps. The results of the experiments show the high performance and low time complexity of the PCMS system. All in all, the PCMS system is a reliable and effective tool for mining parallel corpora from the web.

References

Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., et al. (1990). A statistical approach to machine translation. *Computational Linguistics, 16(2),* 79-85.

Chen, J., Chau, R., and Yeh, C. H. (2004). Discovering parallel text from the World Wide Web. In *Proc. of DMWI-04,* Dunedin, New Zealand.

Landauer, T. K. and Littman, M. L. (1990). Fully automatic cross-language document retrieval using latent semantic indexing. In *Proc. of the 6th Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research,* Waterloo, Ontario.

LDC. (1999). Linguistic Data Consortium (LDC) home page. http://www.ldc.upenn.edu

Ma, X. and Liberman, M. Y. (1999). BITS: A method for bilingual text search over the web. In *Proc. of the Machine Translation Summit VII.*

Oard, D. W. (1997). Cross-language text retrieval research in the USA. In *Proc. of the 3rd ERCIM DELOS Workshop,* Zurich, Switzerland.

Resnik, P. (1998). Parallel strands: A preliminary investigation into mining the web for bilingual text. In *Proc. of AMTA-98,* Langhorne, PA.

Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *Computational Linguistics, 29(3),* 349-380.

Zhang, Y., Wu, K., Gao, J., and Vines, P. (2006). Automatic acquisition of Chinese-English parallel corpus from the web. *In Proceedings of ECIR-06,* London.

Appendix A: A Sample Bilingual Dictionary

abandon --- 背弃
abandon --- 丢弃
abandon --- 放弃
abandon --- 抛弃
abc --- 初步
abc --- 入门
abc --- 字母
abc --- 基本
……
depart --- 出发
depart --- 放弃
depart --- 离开
depart --- 起程
……
leave --- 放弃
leave --- 离开
leave --- 离去
leave --- 留下
……

Fast Duplicate Document Detection using Multi-level Prefix-filter

Kenji Tateishi and **Dai Kusui**
NEC Corporation
Takayama, Ikoma, Nara, 630-0101, Japan
{k-tateishi@bq, kusui@ct}.jp.nec.com

Abstract

Duplicate document detection is the problem of finding all document-pairs rapidly whose similarities are equal to or greater than a given threshold. There is a method proposed recently called prefix-filter that finds document-pairs whose similarities never reach the threshold based on the number of uncommon terms (words/characters) in a document-pair and removes them before similarity calculation. However, prefix-filter cannot decrease the number of similarity calculations sufficiently because it leaves many document-pairs whose similarities are less than the threshold. In this paper, we propose multi-level prefix-filter, which reduces the number of similarity calculations more efficiently and maintains the advantage of prefix-filter (no detection loss, no extra parameter) by applying multiple different prefix-filters.

1 Introduction

Duplicate Document Detection (DDD) is the problem of finding all document-pairs rapidly whose similarities are equal to or greater than a given threshold. DDD is often used for data cleaning of customer databases, trend analysis of failure case databases in contact centers, and can be applied for spam filtering by detecting duplicate blog documents. After receiving target documents and the similarity threshold (ST), the Duplicate Document Detection System (DDDS) shows users all document pairs whose similarities are equal or greater than ST, or document groups these document pairs unify. In the case of data cleaning, DDDS additionally requires users to confirm whether each document pair result is truly duplicated.

The naive implementation of DDD requires similarity calculations of all document pairs, but it demands huge time according to the number of target documents. The current techniques apply the two-stage approach: (i) Reduce document pairs using shallow filtering methods, and then (ii) calculate similarities between the remaining document pairs. Among them, prefix-filter(Sarawagi and Kirpal, 2004)(Chaudhuri et al., 2006)(Bayardo et al., 2007) is a filtering method that finds document-pairs whose similarities never reach the threshold based on the number of uncommon terms (words/characters) in a document-pair, and that removes them before similarity calculation.

For example, suppose that a document pair is composed of 10 terms, and 80% similarity means 8 terms are in common in the document pair. In this case, if the similarity of a document pair is equal to or greater than 80% and 3 terms are selected from one document, the other document must contain at least one of the 3 terms. Therefore, prefix-filter can remove document pairs where one document does not contain any of the 3 terms selected from the other. It can be implemented rapidly by index files. Prefix-filter has two advantages compared with other filtering methods: (i) All document pairs equal to or greater than the similarity threshold (ST) are obtained without any detection loss, and (ii) no extra parameter for filtering is required other than ST.

The problem with prefix-filter is that it cannot reduce similarity calculations sufficiently because it leaves many document-pairs whose similarities are less than ST. Document-pairs that prefix-filter can remove depend on terms selected from each document (in the above example, which 3 terms are selected). At worst, document pairs where only one term is in common might remain. The processing time of DDD can be approximated by the product of the number of similarity calculations and the pro-

cessing time of each similarity calculation. In order to identify the same document pairs correctly, a deep similarity function considering synonyms and variants is essential. Therefore, the number of similarity calculations should decrease as mush as possible.

In this paper, we propose multi-level prefix-filter, which reduces the number of similarity calculations more efficiently and maintains the advantages of prefix-filter (no detection loss, no extra parameter) by applying multiple different prefix-filters. Each prefix-filter chooses terms from each document based on a different priority decision criterion, and removes different document-pairs. It finally calculates the similarities of the document-pairs left by all of the prefix-filters. We conducted an experiment with a customer database composed of address and company name fields, and used edit-similarity for the similarity calculation. The result showed that multi-level prefix-filter could reduce the number of similarity calculations to 1/4 compared with the current prefix-filter.

2 Prefix-filter

Prefix-filter finds document-pairs whose similarities never reach the similarity threshold (ST) based on the number of uncommon terms in a document-pair, and that removes them before the similarity calculation. A DDDS with prefix-filter processes the following four steps. [1]

Step 1: Define x: the minimum proportion of common terms in a document pair whose similarity is equal to or greater than $ST (0 \leq ST \leq 1)$.

Step 2: Decide priorities of all terms on target documents.

Step 3: Select terms from each document according to the priorities in Step 2 until the proportion of selected terms exceeds $1 - x$.

Step 4: Remove document pairs that share no terms selected in Step 3, and calculate the similarities of the remaining document pairs.

Let us illustrate how prefix-filter works briefly. For example, a user inputs 6 documents as in Fig.1 and sets the similarity threshold at $ST = 0.6$ and chooses edit-similarity as the similarity function. Note that edit-similarity between document d1 and document d2, denoted as $edit_sim(d1, d2)$, is defined as follows.

$$edit_sim(d1, d2) = 1 - \frac{edit_distance(d1, d2)}{max(|d1|, |d2|)}$$

Here, $|d1|$ and $|d2|$ denotes the length of $d1$ and $d2$ respectively, and $edit_distance(d1, d2)$ represents the minimum number of edit operations (insertion, deletion, and substitution) that convert $d1$ to $d2$. For example, $edit_distance(d1, d5)$ in Fig.1 is 4: delete E, H, and I, and insert M. Then, $max(|d1|, |d5|)$ is 9, derived from $|d1| = 9$ and $|d5| = 7$. Therefore, $edit_sim(d1, d5) = 1 - (4/9) = 0.45$.

In the first step, when the similarity function is edit-similarity, the minimum proportion of common terms (characters) in a document pair whose similarity is equal or greater than $ST = 0.6$ is $x = 0.6$. This means the similarity of a document pair in which the proportion of common terms is less than 0.6 never reaches 0.6. x can be derived from the similarity function (see Appendix A).

In step 2, DDDS decides the priorities of all terms on target documents. Fig. 1 (a) gives all terms contained in the 6 documents priorities from the lowest document frequency (if the same frequency, alphabetical order). Regardless of the priority decision criteria, the similarities of document pairs removed are always less than ST, but document pairs removed differ. Empirically, it is known that giving high priority from the term of the lowest frequency is effective because the lower the frequency of a term, the lower the probability of a document pair containing that term(Chaudhuri et al., 2006).

In step 3, DDDS chooses terms from each document according to the priority decision criterion of step 2 in Fig.1 (a) until the proportion of selected terms exceeds $1 - x = 0.4$. For example, the proportion is over 0.4 when DDDS selects 4 terms from d1, composed of 9 terms. DDDS selects 4 terms according to (a): $\{A, B, C, I\}$. Fig.1 (b) shows selected terms using boldface and background color.

Finally, DDDS removes document pairs that share no terms selected in step 3, and calculates similarities of the remaining document pairs. The similarities of document pairs with no common terms never

[1] Here, we show the simplest prefix-filter of (Chaudhuri et al., 2006)

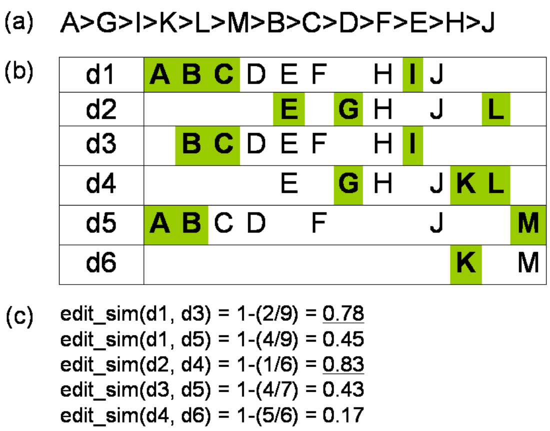


Figure 1: Overview of prefix-filter.

reach 0.6 because the proportion of common terms is less than 0.6. Prefix-filter can be implemented easily using an index file, storing the relation of each selected term and the list of document IDs including the term. As a result, document d1 targets d3 and d5 on similarity calculation. Finally, the number of similarity calculations can be reduced by 5 times while naive solution requires (6*5)/2=15 times.

3 Multi-level prefix-filter

The problem with prefix-filter is that it cannot reduce similarity calculations sufficiently because it leaves many document-pairs whose similarities are less than ST. Document-pairs that prefix-filter can remove depend on terms selected from each document. At worst, document pairs where only one term is in common might remain. In the case of selecting terms according to priority decision criterion (a) in Fig.1, for example, a document pair {d4,d6} on (b) remains although only K is in common. In order to identify the same document pairs correctly, a deep similarity function such as edit-similarity is essential. Therefore, the number of similarity calculations should be decreased as much as possible.

We propose multi-level prefix-filter, which reduces the number of similarity calculations more efficiently by applying multiple different prefix-filters. Each prefix-filter chooses terms from each document based on different priority decision criteria, and removes different document-pairs. It finally calculates the similarities of document-pairs left by all of the prefix-filters. That is why multi-level prefix-filter can reduce the number of document pairs more comprehensively than the current prefix-filter (without any detection loss). Fig.2 illustrates an example of multi-level prefix-filter, applying prefix-filter twice. After DDDS changes priority decision criterion between the first and second prefix-filter, terms selected from each document vary. As a result, document pairs filtered by each prefix-filter change as well. The product of document pairs each prefix-filter leaves leads to the reduction of similarity calculations by 3 times.

Let us explain two kinds of priority decision criteria of terms in the following sections.

3.1 Priority decision using $Score(n, w)$

We define Score(n,w), the score of a term w on n-th prefix-filter, as follows, and give a higher priority to a smaller value of Score(n,w).

$$Score(n, w) = \begin{cases} df(w) & n = 1 \\ 0.1 * df(w) + \displaystyle\sum_{i=1}^{n-1} sdf(i, w) & n \geq 2 \end{cases}$$

where $df(w)$ is the document frequency of w over the target documents, and $sdf(i, w)$ denotes the number of documents in which w was selected on i-th prefix-filter. The basic concept is to give a higher priority to a term of smaller frequency. As mentioned before, this is effective because the lower the frequency of a term, the lower the probability of a document pair containing that term. On the other hand, it is expected that a multi-level prefix-filter becomes more effective if each prefix-filter can filter different document pairs. Therefore, after the second prefix-filter ($n \geq 2$), we give a higher priority to a term whose frequency is small (first term) and which was not selected by previous prefix-filters (second term).

Fig.3 illustrates the process of multi-level prefix-filter based on this creterion. This multi-level prefix-filter can be implimented using two kinds of index files (W_INDEX, D_INDEX) rapidly. If PC with multiple processers, it is easy to parallelize filtering process.

3.2 Priority decision using $Score(d, n, w)$

We define $Score(d, n, w)$, the score of a term w contained in document d on n-th prefix-filter, as fol-

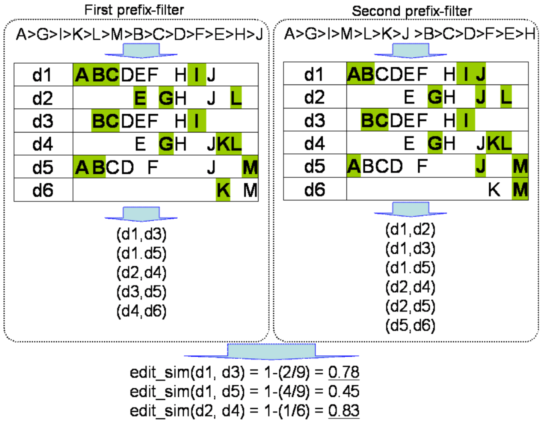


Figure 2: Overview of multi-level prefix-filter.

lows, and give a higher priority to a smaller value of $Score(d, n, w)$.

$$Score(d, n, w) = \begin{cases} df(w) & n = 1 \\ |DS^d_{n-1} \cap DSS_w| & n \geq 2 \end{cases}$$

where DS^d_{n-1} is target documents of similarity calculation of d left after the $n-1$-th prefix-filter, and DSS_w is documents containing a term w. The basic concept is to give a higher priority to a term that can filter many document pairs. It decides the priorities of terms on n-th prefix-filter after waiting for the result of $n-1$-th prefix-filter.

4 Experiments

4.1 Experimental method

We compared multi-level prefix-filter with the current prefix-filter in order to clarify how much the proposed method could reduce the number of similarity calculations. We used a customer database in Japanese, composed of 200,000 records, and had been used for data cleaning. Each record has two fields, company name and address, averaging 11 terms and 18 terms, respectively. We selected edit-similarity as the similarity function, and set 80% as ST. The database contains 86031 (43%) duplicated documents (records) in the company name, and 123068 (60%) in the address field when we assumed document pairs whose similarity was equal to or greater than 80%. A DDDS with multi-level prefix-filter ran on an NEC Express 5800 with Windows 2000, 2.6GHz Pentium Xeon and 3.4 GByte of memory.

```
N: Number of applying prefix-filter, D: Target documents, ST: Similarity Threshold
Index creation process:
for(w∈D)
        Score(1,w) = df(w)
end for
for(i=1; i≦N; i++)
        for(j=0; j≦|D|; j++)
                W= terms chosen from w∈di of the smallest Score(i,w)
                        until the proportion of selected terms exceeds 1-x.
                for(w∈W)
                        push(D_INDEX(i,dj), w)
                        push(W_INDEX(i,w), dj)
                end for
        end for
        for(w∈D)
                Score(i+1,w)= 0.1 * df(w) + Σ1≦k≦i sdf(k,w)
        end for
end for
Matching process:
for(i=0; i≦|D|;i++)
        Filtering process:
        DS = D
        for(j=1; j≦N; j++) {
                for(w∈W_INDEX(j,di))
                        DSSw ={dk | dk ∈ W_INDEX(j,w), k > i}
                        DSj = DSSw ∪ DSj
                end for
                DS = DS ∩ DSj
        end for
        Similarity calculation process:
        for(ds∈DS)
                push(RESULT, {d,ds}) if (sim(d, ds)≧ST)
        end for
end for
```

Figure 3: Multi-level prefix-filter with Score(n,w).

4.2 Experimental result

Fig.4 (a) shows the comparison between multi-level prefix-filter using $Score(d, n, w)$ and $Score(n, w)$ under the condition that the number of prefix-filters is one or two. The company name field was used for target documents. Although multi-level prefix-filter using Score(n,w) succeeded in the reduction of processing time, Score(d,n,w) failed because of too many score calculations. Therefore, we used $Score(n, w)$ in the following experiments.

Fig.4 (b) shows the number of similarity calculations when the number of applied prefix-filters varies. In this figure, $n = 1$ means the current prefix-filter. The number of similarity calculations decreased most sharply in the case of applying prefix-filters twice on both the company name and address fields, and converged in 10 times. Multi-level prefix-filter reduced the number of similarity calculations by 10 times, about to 1/4 (77% reduction) in the company name field, and about to 1/3 (69% reduction) in the address field.

Fig.4 (c) shows total processing time when the

number of applied prefix-filters varies. It represents the sum of index creation/filtering time and similarity calculation time. When the number of applied prefix-filters increased, the latter decreased because the number of similarity calculations also decreased, but the former increased instead. Note that we did not parallelize the filtering process here. Total processing time decreased most sharply in the case of applying prefix-filters 4 times on both the company name (to be 43%) and address fields (to be 49%).

Fig.4 (d) shows the reduction rate of the number of similarity calculations and processing time when prefix-filter was applied 4 times and the size of target document sets varied. Here, the reduction rate denotes the proportion of the number of similarity calculations or processing time of multi-level prefix-filter, applying prefix-filter 4 times, to those of the current prefix-filter, applying prefix-filter once. This result reveals the effectiveness of multi-level prefix-filter does not change for the size of the target document set.

4.3 Discussion

The experimental results indicated that multi-level prefix-filter could reduce the number of similarity calculations up to 1/4, and that this effectiveness was not lost by changing the size of the target database. In addition, it showed that the optimal number of applied prefix-filters did not depend on the target field or the size of the target database. Therefore, multi-level prefix-filter proved to be more effective than the current prefix-filter without losing the advantages of the current prefix-filter (no detection loss, no extra parameter).

The experimental results also indicated that the company name field was more effective than the address field. As mentioned, the address field was longer than that of the company name field on average, and it contained more duplicated documents. Therefore, we expect that the proposed method is effective in the following situation: (i) the length of each document (record) is short, (ii) the number of duplicate documents has been reduced beforehand by simple filtering methods such as deleting exact match documents or documents different only in space, and (iii) detecting the remaining duplicate documents by using a deep similarity function such as edit-similarity.

5 Related work

Duplicate Document Detection for databases has been researched for a long time(Elmagarmid et al., 2007). The current techniques apply the two-stage approach: (i) Reduce document pairs using shallow filtering methods, and then (ii) calculate similarity between the remaining document pairs. Multi-level prefix-filter belongs to the first step (i).

Current filtering methods were independent of the similarity function. Jaro(Jaro, 1989) proposed Standard Blocking, which created many record blocks in which each record shared the same first n terms, and calculated the similarity of document-pairs included in the same record block. Hernandez(Hernandez and Stolfo, 1995) proposed the Sorted Neighborhood Method (SNM), which first sorted records by a given key function, and then grouped adjacent records within the given window size as a block. McCallum(McCallum et al., 2000) improved them by allowing a record to locate in plural blocks in order to avoid detection loss.

However, the problems of these filtering methods using blocking are that the user needs trial and error parameters such as first n terms for Standard Blocking, and that these incur detection loss in spite of improvements being attempted, caused by two documents of a correct document pair existing in different blocks. Prefix-filter solved these problems: (i) all document pairs equal or more than similarity threshold (ST) are obtained without any detection loss, and (ii) any extra parameter for filtering is not required other than ST. As we clarified in Section 4, multi-level prefix-filter proved to be more effective than the current prefix-filter without losing these advantages.

Another filtering method without any detection loss, called PARTENUM, has been proposed recently(Arasu et al., 2006). However, it needs to adjust two kinds of parameters ($n1$, $n2$) for obtaining optimal processing time according to the size of target document set or the similarity threshold.

6 Conclusion

In this paper, we proposed multi-level prefix-filter, which reduces the number of similarity calculations more efficiently and maintains the advantage of the current prefix-filter by applying multiple different

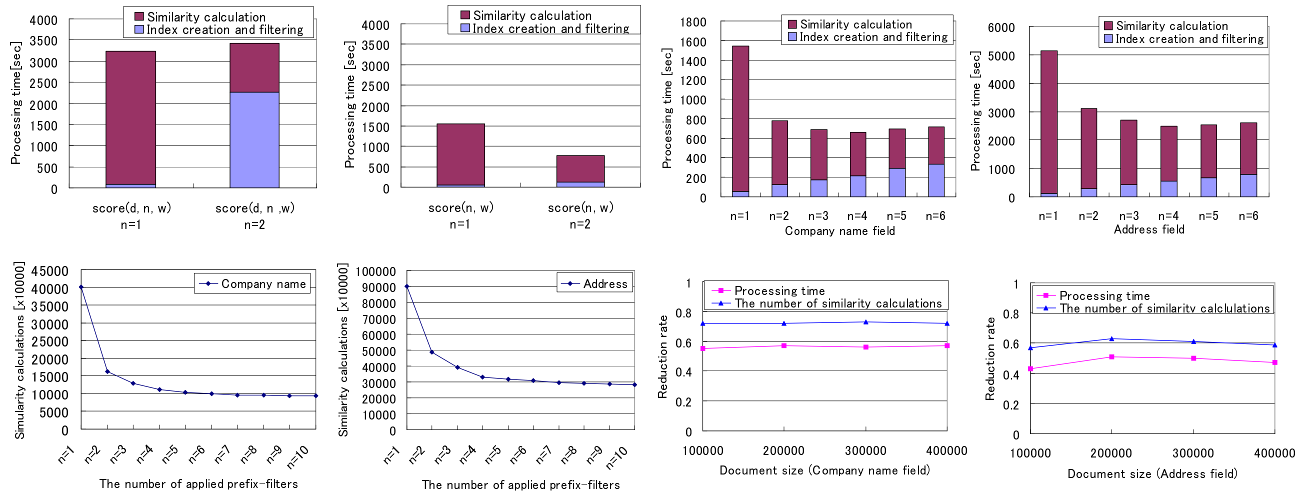


Figure 4: Experimental result.

prefix-filters. Experiments with a customer database composed of 200,000 documents and edit-distance for similarity calculation showed that it could reduce the number of similarity calculations to 1/4 compared with the current prefix-filter.

References

Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. 2006. Efficient exact set-similarity joins. *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 918–929.

Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. 2007. Scaling up all pairs similarity search. *Proceedings of the 16th International Conference on World Wide Web*, pages 131–140.

Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. 2006. A primitive operator for similarity joins in data cleaning. *Proceedings of the 22nd International Conference on Data Engineering(ICDE'06)*, pages 5–16.

Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering, vol.19, no.1*, pages 1–15.

Mauricio A. Hernandez and Salvatore J. Stolfo. 1995. The merge/purge problem for large databases. *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 127–138.

M. A. Jaro. 1989. Advances in record linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Society, 84 (406)*, pages 414–420.

Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178.

Sunita Sarawagi and Alok Kirpal. 2004. Efficient set joins on similarity predicates. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 743–754.

A The minimum proportion of common terms

Here, we explain how to obtain x of edit-similarity. First,

$$edit_distance(d1, d2) \geq max(|d1|, |d2|) - |d1 \cap d2|$$

($|d1 \cap d2|$ denotes the number of common terms in both $d1$ and $d2$), and

$$\begin{aligned} ST &\leq edit_sim(d1, d2) \leq \frac{|d1 \cap d2|}{max(|d1|, |d2|)} \\ &\leq \frac{|d1 \cap d2|}{|d1|}. \end{aligned}$$

Therefore,

$$x = min\{\frac{|d1 \cap d2|}{|d1|}\} = ST.$$

TOWARDS DATA AND GOAL ORIENTED ANALYSIS: TOOL INTER-OPERABILITY AND COMBINATORIAL COMPARISON

Yoshinobu Kano[1] **Ngan Nguyen**[1] **Rune Sætre**[1] **Kazuhiro Yoshida**[1]
Keiichiro Fukamachi[1] **Yusuke Miyao**[1] **Yoshimasa Tsuruoka**[3]
Sophia Ananiadou[2,3] **Jun'ichi Tsujii**[1,2,3]

[1]Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Tokyo

[2]School of Computer Science, University of Manchester
PO Box 88, Sackville St, MANCHESTER M60 1QD, UK

[3]NaCTeM (National Centre for Text Mining), Manchester Interdisciplinary Biocentre, University of Manchester, 131 Princess St, MANCHESTER M1 7DN, UK

{kano,nltngan,satre,kyoshida,keif,yusuke,tsujii}@is.s.u-tokyo.ac.jp
{yoshimasa.tsuruoka,sophia.ananiadou}@manchester.ac.uk

Abstract

Recently, NLP researches have advanced using F-scores, precisions, and recalls with gold standard data as evaluation measures. However, such evaluations cannot capture the different behaviors of varying NLP tools or the different behaviors of a NLP tool that depends on the data and domain in which it works. Because an increasing number of tools are available nowadays, it has become increasingly important to grasp these behavioral differences, in order to select a suitable set of tools, which forms a complex workflow for a specific purpose. In order to observe such differences, we need to integrate available combinations of tools into a workflow and to compare the combinatorial results. Although generic frameworks like UIMA (Unstructured Information Management Architecture) provide interoperability to solve this problem, the solution they provide is only partial. In order for truly interoperable toolkits to become a reality, we also need sharable and comparable type systems with an automatic combinatorial comparison generator, which would allow systematic comparisons of available tools. In this paper, we describe such an environment, which we developed based on UIMA, and we show its feasibility through an example of a protein-protein interaction (PPI) extraction system.

1 Introduction

Recently, an increasing number of TM/NLP tools such as part-of-speech (POS) taggers (Tsuruoka et al., 2005), named entity recognizers (NERs) (Settles, 2005) syntactic parsers (Hara et al., 2005) and relation or event extractors (ERs) have been developed. Nevertheless, it is still very difficult to integrate independently developed tools into an aggregated application that achieves a specific task. The difficulties are caused not only by differences in programming platforms and different input/output data formats, but also by the lack of higher level interoperability among modules developed by different groups.

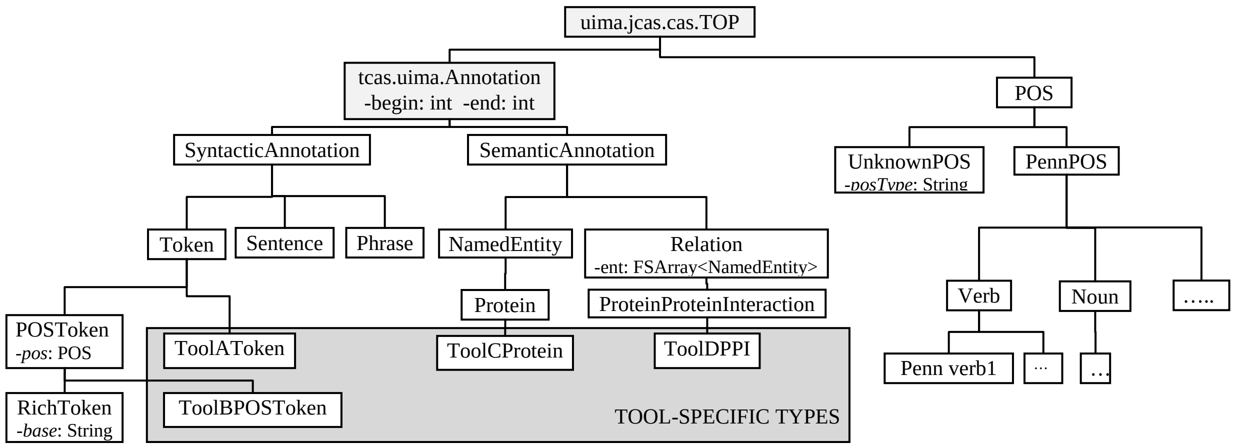


Figure 1. Part of our type system

UIMA, Unstructured Information Management Architecture (Lally and Ferrucci, 2004), which was originally developed by IBM and has recently become an open project in OASIS and Apache, provides a promising framework for tool integration. Although it has a set of useful functionalities, UIMA only provides a generic framework, thus it requires a user community to develop their own platforms with a set of actual software modules. A few attempts have already been made to establish platforms, e.g. the CMU UIMA component repository [1], GATE (Cunningham et al., 2002) with its UIMA interoperability layer, etc.

However, simply wrapping existing modules to be UIMA compliant does not offer a complete solution. Most of TM/NLP tasks are composite in nature, and can only be solved by combining several modules. Users need to test a large number of combinations of tools in order to pick the most suitable combination for their specific task.

Although *types* and *type systems* are the only way to represent meanings in the UIMA framework, UIMA does not provide any specific *types*, except for a few purely primitive *types*. In this paper, we propose a way to design sharable *type systems*. A sharable *type system* designed in this way can provide the interoperability between independently developed tools with fewer losses in information, thus allowing for the combinations of tools and comparisons on these combinations.

We show how our automatic comparison generator works based on a *type system* designed in that way. Taking the extraction of protein-protein interaction (PPI) as a typical example of a composite task, we illustrate how our platform helps users to observe the differences between tools and to construct a system for their own needs.

2 Motivation and Background

2.1 Goal and Data Oriented Evaluation, Module Selection and Inter-operability

There are standard evaluation metrics for NLP modules such as precision, recall and F-value. For basic tasks like sentence splitting, POS tagging, and named-entity recognition, these metrics can be estimated using existing gold-standard test sets.

Conversely, accuracy measurements based on the standard test sets are sometimes deceptive, since its accuracy may change significantly in practice, depending on the types of text and the actual tasks at hand. Because these accuracy metrics do not take into account the importance of the different types of errors to any particular application, the practical utility of two systems with seemingly similar levels of accuracy may in fact differ significantly. To users and developers alike, a detailed examination of how systems perform (on the text they would like to process) is often more important than standard metrics and test sets. Naturally, far greater weight is placed in measuring the end-to-end performance of a composite system than in measuring the performance of the individual components.

In reality, because the selection of modules usually affects the performance of the entire system, it is crucial to carefully select modules that are appropriate for a given task. This is the main reason for having a collection of interoperable

[1] http://uima.lti.cs.cmu.edu/

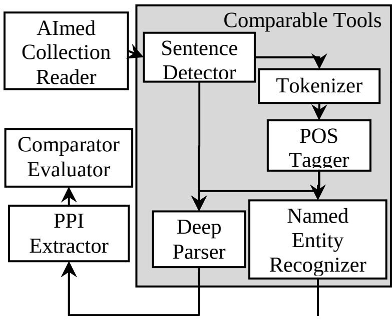


Figure 2. PPI system workflow (conceptual)

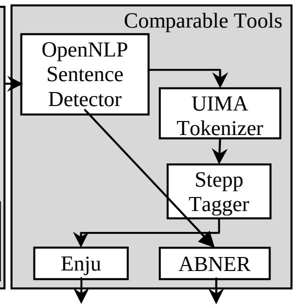


Figure 3. Basic example pattern

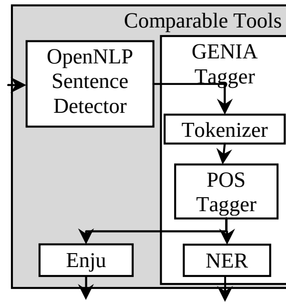


Figure 4. Complex tool example

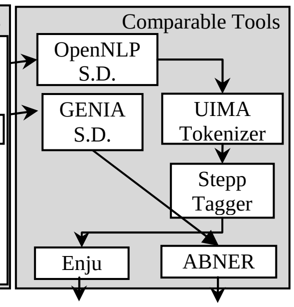


Figure 5. Branch flow pattern

modules. We need to show how the ultimate performance will be affected by the selection of different modules and show the best combination of modules in terms of the performance of the whole aggregated system for the task at hand.

Since the number of possible combinations of component modules is typically large, the system has to be able to enumerate and execute them semi-automatically. This requires a higher level of interoperability of individual modules than just wrapping them for UIMA.

2.2 UIMA

2.2.1 CAS and Type System

The UIMA framework uses the "stand-off annotation" style (Ferrucci et al., 2006). The raw text in a document is kept unchanged during the analysis process, and when the processing of the text is performed, the result is added as new stand-off annotations with references to their positions in the raw text. A Common Analysis Structure (CAS) maintains a set of these annotations, which in itself are objects. The annotation objects in a CAS belong to *types* that are defined separately in a hierarchical *type system*. The features of an *annotation*[2] object have values that are typed as well.

2.2.2 Component and Capability

Each UIMA Component has the *capability* property which describes what types of objects the component may take as the input and what types of objects it produces as the output. For example, a named entity recognizer detects named entities in the text and outputs annotation objects of the type `NamedEntity`.

It is possible to deploy any UIMA component as a SOAP web service, so that we can combine a remote component on a web service with the local component freely inside a UIMA-based system.

3 Integration Platform and Comparators

3.1 Sharable and Comparable Type System

Although UIMA provides a set of useful functionalities for an integration platform of TM/NLP tools, users still have to develop the actual platform by using these functionalities effectively. There are several decisions for the designer to make an integration platform.

Determining how to use *types* in UIMA is a crucial decision. Our decision is to keep different *type systems* by individual groups as they are, if necessary; we require that individual *type systems* have to be related through a sharable *type system*, which our platform defines. Such a shared *type system* can bridge modules with different *type systems*, though the bridging module may lose some information during the translation process.

Whether such a sharable *type system* can be defined or not is dependent on the nature of each problem. For example, a sharable *type system* for POS tags in English can be defined rather easily, since most of POS-related modules (such as POS taggers, shallow parsers, etc.) more or less follow the well established types defined by the Penn Treebank (Marcus et al., 1993) tag set.

Figure 1 shows a part of our sharable *type system*. We deliberately define a highly organized type hierarchy as described above.

Secondly we should consider that the *type system* may be used to compare a similar sort of tools. *Types* should be defined in a distinct and

[2] In the UIMA framework, `Annotation` is a base *type* which has *begin* and *end* offset values. In this paper we call any objects (any subtype of `TOP`) as *annotations*.

hierarchical manner. For example, both tokenizers and POS taggers output an object of *type* `Token`, but their roles are different when we assume a cascaded pipeline. We defined `Token` as a supertvpe, `POSToken` as subtypes of `Token`. Each tool should have an individual *type* to make clear which tool generated which instance, because each tool may have a slightly different definition. This is important because the *capabilities* are represented by these *types*, and the *capabilities* are the only attributes which are machine readable.

3.2 General Combinatorial Comparison Generator

Even if the *type system* is defined in the previously described way, there are still some issues to consider when comparing tools. We illustrate these issues using the PPI *workflow* that we utilized in our experiments.

Figure 2 conceptually shows the *workflow* of our whole PPI system. If we can prepare two or more components for some type of the components in the *workflow* (e.g. two sentence detectors and three POS taggers), then we can make combinations of these tools to form a multiplied number of *workflow* patterns (2x3 = 6 patterns). See Table 1 for the details of UIMA components used in our experiments.

We made a pattern expansion mechanism which generates possible *workflow* patterns automatically from a user-defined *comparable workflow*. A *comparable workflow* is a special *workflow* that explicitly specifies which set of components should be compared. Then, users just need to group comparable components (e.g. ABNER[3] and MedT-NER as a comparable NER group) without making any modifications to the original UIMA components. This aggregation of comparable components is controlled by our *custom workflow controller*.

In some cases, a single tool can play two or more roles (e.g. the GENIA Tagger performs tokenization, POS tagging, and NER; see Figure 4). It may be possible to decompose the original tool into single roles, but in most cases it is difficult and unnatural to decompose such a complex tool. We designed our comparator to detect possible input combinations automatically by the *types* of previously generated *annotations*, and the input *capability* of each posterior component. As described in the previous section, the component should have appropriate *capabilities* with proper *types* in order to permit this detection.

When a component requires two or more input *types* (e.g. our PPI extractor requires outputs of a deep parser and a protein NER system), there could be different components used in the prior flow (e.g. OpenNLP and GENIA sentence detectors in Figure 5). Our comparator also calculates such cases automatically.

	O	U	G	A
G	0	0	-	85
U	86	-	0	7
A	6	6	60	-
O	-	81	0	7

Table 2. `Sentence` comparisons (%).

	OO	UO	GO
UU	89/75	89/75	88/70
GU	89/75	89/75	88/70
GG	92/95	91/95	97/95
OG	100/100	99/99	100/94

Table 3. Part of `Token` comparisons, precision/recall (%).

	OOO	UOS	GOO
UUO	87/74	81/68	85/68
GUG	74/65	73/65	78/65
GGO	92/95	81/84	97/95
OGO	100/100	89/88	100/94

Table 4. Part of `POSToken` comparisons, precision/recall (%)

4 Experiments and Results

We have performed experiments using our PPI extraction system as an example (Kano et al., 2008). It is similar to our BioCreative PPI system (Sætre et al., 2006) but differs in that we have deconstructed the original system into seven different components (Figure 2).

As summarized in Table 1, we have several comparable components and the AImed corpus as the gold standard data. In this case, possible combination *workflow* patterns are `POSToken` for 36, `PPI` for 589, etc.

Table 2, 3, 4 and Figure 6 show a part of the comparison result screenshots between these patterns on 20 articles from the AImed corpus. In the tables, abbreviations like "OOG" stands for a workflow of O(`Sentence`) -> O(`Token`) -

[3] In the example figures, ABNER requires `Sentence` to make the explanation clearer, though ABNER does not require it in actual usage.

G(POSToken), where O stands for OpenNLP, G stands for Genia, U stands for UIMA, etc.

When neither of the compared results include the gold standard data (AImed in this case), the comparison results show a *similarity* of the tools for this specific task and data, rather than an evaluation. Even if we lack an annotated corpus, it is possible to run the tools and compare the results in order to understand the characteristics of the tools depending on the corpus and the tool combinations.

Although the comparison on Sentences shows low scores of *similarities*, Tokens are almost the same; it means that input sentence boundaries do not affect tokenizations so much. POSToken *similarities* drop approximately 0-10%

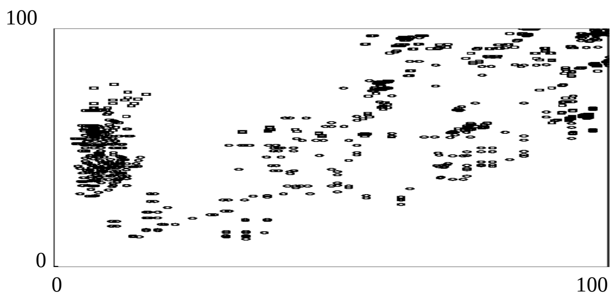


Figure 6. NER (Protein) comparison distribution of precisions (x-axis, %) and recalls (y-axis, %).

from the *similarities* in Token; the differences in Token are mainly apostrophes and punctuations; POSTokens are different because each POS tagger uses a slightly different set of tags: normal Penn tagset for Stepp tagger, BioPenn tagset (includes new tags for hyphenation) for GENIA tagger, and an original apostrophe tag for OpenNLP tagger.

5 Conclusion and Future Work

NLP tasks typically consist of many components, and it is necessary to show which set of tools are most suitable for each specific task and data. Although UIMA provides a general framework with much functionality for interoperability, we still need to build an environment that enables the combinations and comparisons of tools for a specific task.

The *type system* design, which the UIMA framework does not provide, is one of the most critical issues on interoperability. We have thus proposed a way to design a sharable and comparable *type system*. Such a *type system* allows for the automatic combinations of any UIMA compliant components and for the comparisons of these combinations, when the components have proper *capabilities* within the *type system*. We are

Sentence	Token	POSToken	RichToken	Protein	Phrase	PPI
GENIA Tagger: Trained on the WSJ, GENIA and PennBioIE corpora (POS). Uses Maximum Entropy (Berger et al., 1996) classification, trained on JNLPBA (Kim et al., 2004) (NER). Trained on GENIA corpus (Sentence Splitter).						
Enju: HPSG parser with predicate argument structures as well as phrase structures. Although trained with Penn Treebank, it can compute accurate analyses of biomedical texts owing to its method for domain adaptation (Hara et al., 2005).						
STePP Tagger: Based on probabilistic models, tuned to biomedical text trained by WSJ, GENIA (Kim et al., 2003) and PennBioIE corpora.						
MedT-NER: Statistical recognizer trained on the JNLPBA data.						
ABNER: From the University of Wisconsin (Settles, 2005), wrapped by the Center for Computational Pharmacology at the University of Colorado.						
Akane++: A new version of the AKANE system (Yakushiji, 2006), trained with SVMlight-TK (Joachims, 1999; Bunescu and Mooney, 2006; Moschitti, 2006) and the AImed Corpus.						
UIMA Examples: Provided in the Apache UIMA example. Sentence Splitter and Tokenizer.						
OpenNLP Tools: Part of the OpenNLP project (http://opennlp.sourceforge.net/), from Apache UIMA examples.						
AImed Corpus: 225 Medline abstracts with proteins and PPIs annotated (Bunescu and Mooney, 2006).						

Legend: Input type(s) required for that tool; Input type(s) required optionally; Output type(s)

Table 1. List of UIMA Components used in our experiment.

preparing to make a portion of the components and services described in this paper publicly available (http://www-tsujii.is.s.u-tokyo.ac.jp/uima/).

The final system shows which combination of components has the best score, and also generates comparative results. This helps users to grasp the characteristics and differences among tools, which cannot be easily observed by the widely used F-score evaluations only.

Future directions for this work includes combining the output of several modules of the same kind (such as NERs) to obtain better results, collecting other tools developed by other groups using the sharable *type system*, making machine learning tools UIMA compliant, and making grid computing available with UIMA workflows to increase the entire performance without modifying the original UIMA components.

Acknowledgments

We wish to thank Dr. Lawrence Hunter's text mining group at the Center for Computational Pharmacology for discussing with us and making their tools available for this research. This work was partially supported by NaCTeM (the UK National Centre for Text Mining), Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Genome Network Project (MEXT, Japan). NaCTeM is jointly funded by JISC/BBSRC/EPSRC.

References

Berger, Adam L., Vincent J. Della Pietra, and Stephen A. Della Pietra. "A maximum entropy approach to natural language processing." *Comput. Linguist.* (MIT Press) 22, no. 1 (1996): 39-71.

Bunescu, Razvan, and Raymond Mooney. "Subsequence Kernels for Relation Extraction." Edited by Weiss Y., Scholkopf B. and Platt J., 171-178. Cambridge, MA: MIT Press, (2006).

Cunningham, H., D. Maynard, K. Bontcheva, and V. Tablan. "GATE: A framework and graphical development environment for robust NLP tools and applications." *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics.* (2002).

Ferrucci, David, Adam Lally, Daniel Gruhl, and Edward Epstein. "Towards an Interoperability Standard for Text and Multi-Modal Analytics." IBM Research Report, RC24122. (2006).

Hara, Tadayoshi, Yusuke Miyao, and Jun'ichi Tsujii. "Adapting a probabilistic disambiguation model of an HPSG parser to a new domain." Edited by Dale Robert, Wong Kam-Fai, Su Jian and Yee Oi. *Natural Language Processing IJCNLP 2005.* Jeju Island, Korea: Springer-Verlag, (2005). 199-210.

Joachims, Thorsten. "Making large-scale support vector machine learning practical." MIT Press, (1999): 169-184.

Kano, Yoshinobu, et al. "Filling the gaps between tools and user: a tool comparator, using protein-protein interaction as an example." *Proceedings of The Pacific Symposium on Biocomputing (PSB).* Hawaii, USA, To appear, (2008).

Kim, Jin-Dong, Tomoko Ohta, Yoshimasa Tsuruoka, and Yuka Tateisi. "Introduction to the Bio-Entity Recognition Task at JNLPBA." *Proceedings of the International Workshop on Natural Language Processing.* Geneva, Switzerland, (2004). 70-75.

Kim, Jin-Dong, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. "GENIA corpus - a semantically annotated corpus for bio-textmining." *Bioinformatics* (Oxford University Press) 19, no. suppl. 1 (2003): i180-i182.

Lally, Adam, and David Ferrucci. "Building an Example Application with the Unstructured Information Management Architecture." *IBM Systems Journal* 43, no. 3 (2004): 455-475.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. "Building a Large Annotated Corpus of English: The Penn Treebank." *Computational Linguistics* 19, no. 2 (1993): 313-330.

Moschitti, Alessandro. "Making Tree Kernels Practical for Natural Language Learning." *EACL.* (2006).

Sætre, Rune, Kazuhiro Yoshida, Akane Yakushiji, Yusuke Miyao, Yuichiroh Matsubayashi, and Tomoko Ohta. "AKANE System: Protein-Protein Interaction Pairs in BioCreAtIvE2 Challenge." *Proceedings of the Second BioCreative Challenge Evaluation Workshop.* (2007).

Settles, B. "ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text." *Bioinformatics* (Oxford University Press) 21, no. 14 (2005): 3191-3192.

Tsuruoka, Yoshimasa, Yuka Tateishi, Jin-Dong Kim, and Tomoko Ohta. "Developing a Robust Part-of-Speech Tagger for Biomedical Text." *Advances in Informatics - 10th Panhellenic Conference on Informatics.* Volos, Greece, (2005). 382-392.

Yakushiji, Akane. "Relation Information Extraction Using Deep Syntactic Analysis." PhD Thesis, University of Tokyo, (2006).

A Co-occurrence Graph-based Approach for Personal Name Alias Extraction from Anchor Texts

Danushka Bollegala *
The University of Tokyo
7-3-1, Hongo, Tokyo,
113-8656, Japan
danushka@mi.ci.i.u-tokyo.ac.jp

Yutaka Matsuo
National Institute of Advanced Industrial Science and Technology
1-18-13, Sotokanda, Tokyo,
101-0021, Japan
y.matsuo@aist.go.jp

Mitsuru Ishizuka
The University of Tokyo
7-3-1, Hongo, Tokyo,
113-8656, Japan
ishizuka@i.u-tokyo.ac.jp

Abstract

A person may have multiple name aliases on the Web. Identifying aliases of a name is important for various tasks such as information retrieval, sentiment analysis and name disambiguation. We introduce the notion of a word co-occurrence graph to represent the mutual relations between words that appear in anchor texts. Words in anchor texts are represented as nodes in the co-occurrence graph and an edge is formed between nodes which link to the same url. For a given personal name, its neighboring nodes in the graph are considered as candidates of its aliases. We formalize alias identification as a problem of ranking nodes in this graph with respect to a given name. We integrate various ranking scores through support vector machines to leverage a robust ranking function and use it to extract aliases for a given name. Experimental results on a dataset of Japanese celebrities show that the proposed method outperforms all baselines, displaying a MRR score of 0.562.

1 Introduction

Searching for information about people in the Web is one of the most common activities of Internet users. Around 30% of search engine queries include person names (Guha and Garg, 2004). However, an individual might have multiple nicknames or *aliases* on the Web. For example, the famous Japanese major league baseball player *Hideki Matsui* is often called as *Godzilla* in web contents. Identifying aliases of a name is important in various tasks such as information retrieval (Salton and McGill, 1986), sentiment analysis (Turney, 2002) and name disambiguation (Bekkerman and McCallum, 2005).

In information retrieval, to improve recall of a web search on a person name, a search engine can automatically expand the query using aliases of the name. In our previous example, a user who searches for *Hideki Matsui* might also be interested in retrieving documents in which Matsui is referred to as *Godzilla*. People use different aliases when expressing their opinions about an entity. By aggregating texts written on an individual that use various aliases, a sentiment analysis system can make an informed judgment on the sentiment. Name disambiguation focuses on identifying different individuals with the same name. For example, for the name *Jim Clark*, aside from the two most popular namesakes - the formula-one racing champion and the founder of Netscape - at least ten different people are listed among the top 100 results returned by Google for the name. Although namesakes have identical names, their nicknames usually differ. Therefore, a name disambiguation algorithm can benefit from the knowledge related to name aliases.

We propose an alias extraction method that exploits anchor texts and the links indicated by the anchor texts. Link structure has been studied extensively in information retrieval and has been found to be useful in various tasks such as ranking of web pages, identification of hub-authority

* Research Fellow of the Japan Society for the Promotion of Science (JSPS)

sites, text categorization and social network extraction (Chakrabarti, 2003). Anchor texts pointing to an url provide useful semantic clues regarding the resource represented by the url.

If the majority of inbound anchor texts of an url contain a person name, then it is likely that the remainder of the anchor texts contain information about aliases of the name. For example, an image of *Hideki Matsui* on a web page might be linked using the real name, *Hideki Matsui*, as well as aliases *Godzilla* and *Matsu Hide*. However, extracting aliases from anchor texts is a challenging problem due to the noise in both link structure and anchor texts. For example, web pages of extremely diverse topics link to yahoo.com using various anchor texts. Moreover, given the scale of the Web, broken links and incorrectly linked anchor texts are abundant. Naive heuristics are insufficient to extract aliases from anchor texts.

Our main contributions are summarized as follows:

- We introduce **word co-occurrence graphs** to represents words that appear in anchor texts and formalize the problem of alias extraction as a one of ranking nodes in the graph with respect to a given name.

- We define various ranking scores to evaluate the appropriateness of a word as an alias of a name. Moreover, the ranking scores are integrated using support vector machines to leverage a robust alias detection method.

2 Related Work

Hokama and Kitagawa (2006) propose an alias extraction method that is specific to Japanese language. For a given name p, they search for the query * *koto p* [1] and extract the words that match the asterisk. However, *koto* is highly ambiguous and extracts lots of incorrect aliases. Moreover, the method cannot extract aliases when a name and its aliases appear in separate documents.

Anchor texts and link structure have been employed in synonym extraction (Chen et al., 2003) and translations extraction (Lu et al., 2004). Chen et al. (2003) propose the use of hyperlink structure

[1] *koto* is written in hiragana and and means *also known as*

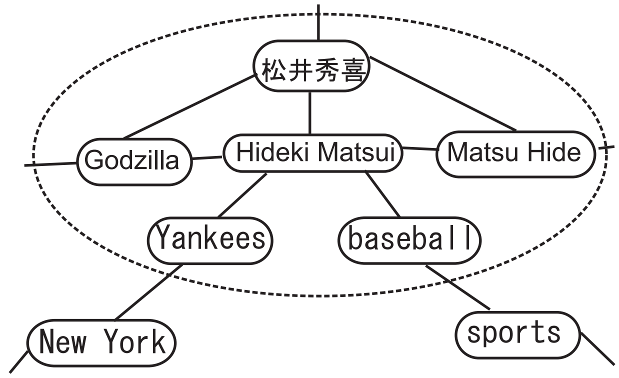


Figure 1: Co-occurrence graph for *Hideki Matsui*

within a particular domain to generate a domain-specific thesaurus. First, a set of high quality websites from a given domain is selected. Second, several link analysis techniques are used to remove noisy links and the navigational structure of the website is converted into a content structure. Third, pointwise mutual information is applied to identify phrases within content structures to create a domain specific thesaurus. They evaluate the thesaurus in a query expansion task. Anchor texts written in different languages that point the same object have been used in cross-language information retrieval (CLIR) to translate user queries. Lu et al. (2004) extend this idea by associating anchor texts written using a pivotal third language to find translations of queries.

3 Method

3.1 Outline

We introduce *word co-occurrence graph*, an undirected graph, to represent words that appear in anchor texts. For each word that appears in the vocabulary of words in anchor texts, we create a node in the graph. Two words are considered as co-occurring if two anchor texts containing these words link to the same url. An edge is formed between two nodes if the words represented by those nodes co-occur. Figure 1 illustrates a portion of the co-occurrence graph in the proximity of *Hideki Matsui* as extracted by this method from anchor texts.

Representing words that appear in anchor texts as a graph enables us to capture the complex inter-relations between the words. Words in inbound anchor texts of an url contain important semantic clues

regarding the resource represented by the url. Such words form a clique in the co-occurrence graph, indicating their close connectivity. Moreover, co-occurrence graphs represent indirect relationships between words. For example, in Figure 1 *Hideki Matsui* is connected to *New York* via *Yankees*.

We model the problem of extracting aliases as a one of ranking nodes in the co-occurrence graph with respect to a real name. Usually, an individual has just one or two aliases. A name alias extraction algorithm must identify the correct aliases among a vast number of related terms for an individual.

3.2 Word Co-occurrence Graph

Let V be the vocabulary of words w_i that appear in anchor texts. The boolean function $A(a_i, w_i)$ returns true if the anchor text a_i contains the word w_i. Moreover, let the boolean function $L(a_i, u_i)$ to be true if the anchor text a_i points to url u_i. Then two words w_i, w_j are defined to be *co-occurring* in a url u, if $A(a_i, w_i) \wedge A(a_j, w_j) \wedge L(a_i, u) \wedge L(a_j, u)$ is true for at least one pair of anchor texts $(a_i,\, a_j)$. In other words, two words are said to co-occur in an url if at least one inbound pair of anchor texts contains the two words. Moreover, we define the number of co-occurrences of w_i and w_j to be the number of different urls they co-occur.

We define *word co-occurrence graph*, $G(V, E)$ (V is the set of nodes and E is the set of edges) as an undirected graph where each word w_i in vocabulary V is represented by a node in the graph. Because one-to-one mapping pertains between a word and a node, for simplicity we use w_i to represent both the word and the corresponding node in the graph. An edge $e_{ij} \in E$ is created between two nodes w_i, w_j if they co-occur. Given a personal name p, represented by a node p in the co-occurrence graph, our objective is to identify the nodes that represent aliases of p. We rank the nodes in the graph with respect to p such that more likely a node is an alias of p, the higher the rank it is assigned. According to our definition, a node that lies n hops away from p has an n-order co-occurrence with p. Considering the fact that a single web page might link to many pages with diverse topics, higher order co-occurrences with p (i.e. nodes that appear further from p) are unreliable as aliases of p. Consequently, we limit $C(p)$, the set of candidate aliases of p, to nodes which are directly connected to p in the graph. In Figure 1 candidates of *Hideki Matsui* fall inside the dotted ellipse.

Table 1: Contingency table for a candidate alias x

	x	$C(p) - \{x\}$	
p	k	$n - k$	n
$V - \{p\}$	$K - k$	$N - n - K + k$	$N - n$
V	K	$N - K$	N

3.3 Ranking of Candidates

To evaluate the strength of co-occurrence between a candidate alias and the real name, for each candidate alias x in $C(p)$ we create a contingency table as shown in Table 1. In Table 1, the first row represents candidates of p and the first column represents nodes in the graph. Therein, k is the number of urls in which p and x co-occur, K is the number of urls in which at least one inbound anchor text contains the candidate x, n is the number of urls in which at least one inbound anchor text contains p and N is the total number of urls in the crawl. Next, we define various ranking scores based on Table 1.

Simplest of all ranking scores is the *link frequency* (lf). We define link frequency of an candidate x as the number of different urls in which x and p co-occur. This is exactly the value of k in Table 1.

Link frequency is biased towards highly frequent words. A word that has a high frequency in anchor texts can also report a high co-occurrence with p. *tfidf* measure which is popularly used in information retrieval can be used to normalize this bias. *tfidf* is computed from Table 1 as follows,

$$tfidf(n_j) = k \log \frac{N}{K+1}.$$

From Table 1 we compute co-occurrence measures; log likelihood ratio **LLR** (Dunning, 1993), chi-squared measure **CS**, point-wise mutual information **PMI** (Church and Hanks, 1991) and hyper geometric distribution **HG** (Hisamitsu and Niwa, 2001). Each of these measures is used to rank candidate aliases of a given name. Because of the limited availability of space, we omit the definitions of these measures.

Furthermore, we define popular set overlap measures; *cosine measure*, *overlap coefficient* and *Dice coefficient* from Table 1 as follows,

$$\text{cosine}(p, x) = \frac{k}{\sqrt{n} + \sqrt{K}},$$

$$\text{overlap}(p, x) = \frac{k}{\min(n, K)},$$

$$\text{Dice}(p, x) = \frac{2k}{n + K}.$$

3.4 Hub weighting

A frequently observed phenomenon on the Web is that many web pages with diverse topics link to so called *hubs* such as Google, Yahoo or Amazon. Because two anchor texts might link to a hub for entirely different reasons, co-occurrences coming from hubs are prone to noise. To overcome the adverse effects of a hub h when computing the ranking scores described in section 3.3, we multiply the number of co-occurrences of words linked to h by a factor $\alpha(h, p)$ where,

$$\alpha(h, p) = \frac{t}{d - 1}. \tag{1}$$

Here, t is the number of inbound anchor texts of h that contain the real name p, d is the total number of inbound anchor texts of h. If many anchor texts that link to h contain p (i.e., larger t value) then the reliability of h as a source of information about p increases. On the other hand, if h has many inbound links (i.e., larger d value) then it is likely to be a noisy hub and gets discounted when multiplied by $\alpha(<< 1)$. Intuitively, Formula 1 boosts hubs that are likely to be containing information regarding p, while penalizing those that contain various other topics.

3.5 Training

In section 3.3 we introduced 9 ranking scores to evaluate the appropriateness of a candidate alias for a given name. Each of the scores is computed with and without weighting for hubs, resulting in $2 \times 9 = 18$ ranking scores. The ranking scores capture different statistical properties of candidates; it is not readily apparent which ranking scores best convey aliases of a name. We use real world name-alias data to learn the proper combination of the ranking scores.

We represent each candidate alias as a vector of the ranking scores. Because we use the 18 ranking scores described above, each candidate is represented by an 18-dimensional vector. Given a set of personal names and their aliases, we model the training process as a preference learning task. For each name, we impose a binary preference constraint between the correct alias and each candidate.

For example, let us assume for a name w_p we selected the four candidates a_1, a_2, a_3, a_4. Without loss of generality, let us further assume that a_1 and a_2 are the correct aliases of p. Therefore, we form four partial preferences: $a_1 \succ a_3$, $a_1 \succ a_4$, $a_2 \succ a_3$ and $a_2 \succ a_4$. Here, $x \succ y$ denotes the fact that x is preferred to y. We use ranking SVMs (Joachims, 2002) to learn a ranking function from preference constraints. Ranking SVMs attempt to minimize the number of discordant pairs during training, thereby improving average precision. The trained SVM model is used to rank a set of candidates extracted for a name. Then the highest ranking candidate is selected as the alias of the name.

4 Experiments

We crawled Japanese web sites and extracted anchor texts and urls linked by the anchor texts. A web site might use links for purely navigational purposes, which convey no semantic clue. To remove navigational links in our dataset, we prepare a list of words that are commonly used in navigational menus, such as *top, last, next, previous, links*, etc and remove anchor texts that contain those words. In addition we remove any links that point to pages within the same site. All urls with only one inbound anchor text are removed from the dataset. After the above mentioned processing, the dataset contains $24,456,871$ anchor texts pointing to $8,023,364$ urls. The average number of inbound anchor texts per url is 3.05 and its standard deviation is 54.02. We tokenize anchor texts using the Japanese morphological analyzer MeCab (Kudo et al., 2004) and select nouns as nodes in the co-occurrence graph.

For training and evaluation purposes we manually assigned aliases for 441 Japanese celebrities. The name-alias dataset covers people from various fields

Table 2: Mean Reciprocal Rank

Method	MRR	Method	MRR
SVM (RBF)	0.5625	lf	0.0839
SVM (Linear)	0.5186	cosine	0.0761
SVM (Quad)	0.4898	tfidf	0.0757
SVM (Cubic)	0.4087	Dice	0.0751
tfidf(h)	0.3957	overlap(h)	0.0750
LLR(h)	0.3879	PMI(h)	0.0624
cosine(h)	0.3701	LLR	0.0604
lf(h)	0.3677	HG	0.0399
HG(h)	0.3297	CS	0.0079
Dice(h)	0.2905	PMI	0.0072
CS(h)	0.1186	overlap	0.0056

of cinema, sports, politics and mass-media. The majority of people in the dataset have only one alias assigned. For each real name in the dataset we extract a set of candidates using the proposed method. We then sort the real names in the dataset according to the number of candidates extracted for them. We select the top 50 real names with the greatest number of candidate aliases for evaluation purposes because recognizing the correct alias from numerous candidates is a more challenging task that enables us to perform a strict evaluation. On average a name in our evaluation dataset has 6500 candidates, of which only one is correct. The rest of the 391 ($441 - 50$) names are used for training.

We compare the proposed method (SVM) against various baseline ranking scores using mean reciprocal rank (MRR) (Baeza-Yates and Ribeiro-Neto, 1999). The MRR is defined as follows;

$$\mathrm{MRR} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{R_i}. \qquad (2)$$

Therein, R_i is the rank assigned to a correct alias and n is the total number of aliases. The MRR is widely used in information retrieval to evaluate the ranking of search results. Formula 2 gives high MRR to ranking scores which assign higher ranks to correct aliases.

Our experimental results are summarized in Table 2. The hub weighted versions of ranking scores are denoted by (h). We trained rank SVMs with linear *SVM (Linear)*, quadratic *SVM (Quad)*, cubic *SVM (Cubic)* and radial basis functions (RBF) *SVM (RBF)* kernels. As shown in Table 2, the proposed SVM-based method has the highest MRR values among all methods compared. The best results are obtained with the RBF kernel (SVM RBF). In fact for 21 out of 50 names in our dataset, SVM (RBF) correctly ranks their aliases at the first rank. Considering the fact that each name has more than 6000 candidate aliases, this is a marked improvement over the baselines. It is noteworthy in Table 2 that the hub-weighted versions of ranking scores outperform the corresponding non-weighted version. This justifies the hub weighting method proposed in section 3.4. The hub-weighted tfidf score (tfidf(h)) has the best MRR among the baseline ranking scores. For polynomial kernels, we observe a drop of precision concomitant with the complexity of the kernel, which occurs as a result of over-fitting.

Table 3 shows the top-three ranked aliases extracted for *Hideki Matsui* by various methods. English translation of words are given within brackets. The correct alias, *Godzilla*, is ranked first by SVM (RBF). Moreover, the correct alias is followed by the last name *Matsui* and his team, *New York Yankees*. In fact, tfidf(h), LLR(h) and lf(h) all have the exact ranking for the top three candidates. *Hide*, which is an abbreviated form of *Hideki*, is ranked second by these measures. However, none contains the alias *Godzilla* among the top three candidates. The non-hub weighted measures tend to include general terms such as *Tokyo*, *Yomiuri* (a popular Japanese newspaper), *Nikkei* (a Japanese business newspaper), and *Tokyo stock exchange*. A close analysis revealed that such general terms frequently co-occur with a name in hubs. Without adjusting the co-occurrences coming from hubs, such terms invariably receive high ranking scores, as shown in Table 3.

Incorrect tokenization of Japanese names is a main source of error. Many aliases are out-of-dictionary (*unknown*) words, which are known to produce incorrect tokenizations in Japanese morphological analyzers. Moreover, a name and its aliases can be written in various scripts: Hiragana, Katanaka, Kanji, Roman and even combinations of multiple scripts. Some foreign names such as *David* even have orthographic variants in Japanese: *da-bid-do* or *de-bid-do*. Failing to recognize the different ways in which a name can be written engenders wrong preference constraints during training.

Table 3: Top ranking candidate aliases for Hideki Matsui

Method	First	Second	Third
SVM (RBF)	ゴジラ (Godzilla)	松井 (Matsui)	ヤンキース (Yankees)
tfidf(h)	松井 (Matsui)	秀 (Hide)	ヤンキース (Yankees)
LLR(h)	松井 (Matsui)	秀 (Hide)	ヤンキース (Yankees)
cosine(h)	松井 (Matsui)	ヤンキース (Yankees)	秀 (Hide)
lf(h)	松井 (Matsui)	秀 (Hide)	ヤンキース (Yankees)
HG(h)	松井 (Matsui)	ヤンキース (Yankees)	秀 (Hide)
Dice(h)	松井 (Matsui)	ヤンキース (Yankees)	秀 (Hide)
CS(h)	松井 (Matsui)	メジャーリーグ (Major league)	プレーヤー (player)
lf	東京 (Tokyo)	読売 (Yomiuri)	日経 (Nikkei)
cosine	読売 (Yomiuri)	東証 (Tokyo stock exchange)	松井 (Matsui)
tfidf	読売 (Yomiuri)	東京 (Tokyo)	東証 (Tokyo stock exchange)
Dice	読売 (Yomiuri)	東証 (Tokyo stock exchange)	松井 (Matsui)
overlap(h)	プレー (play)	ゴジラ (Godzilla)	スタインブレナー (Steinbrenner)
PMI(h)	プレー (play)	ゴジラ (Godzilla)	スタインブレナー (Steinbrenner)
LLR	読売 (Yomiuri)	東証 (Tokyo stock exchange)	時事通信社 (jiji.com)
HG	読売 (Yomiuri)	東証 (Tokyo stock exchange)	松井 (Matsui)
CS	時事通信社 (jiji.com)	東証 (Tokyo stock exchange)	読売 (Yomiuri)
PMI	コムタチン (Komdatzien)	写真 (picture)	コンテンツ (contents)
overlap	コムタチン (Komdatzien)	写真 (picture)	コンテンツ (contents)

5 Conclusion

We proposed a method to extract aliases of a given name using anchor texts and link structure. We created a co-occurrence graph to represent words in anchor texts and modeled the problem of alias extraction as a one of ranking nodes in this graph with respect to a given name. In future, we intend to apply the proposed method to extract aliases for other entity types such as products, organizations and locations.

References

R.A. Baeza-Yates and B.A. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press / Addison-Wesley.

R. Bekkerman and A. McCallum. 2005. Disambiguating web appearances of people in a social network. In *Proc. of the World Wide Web Conference (WWW' 05)*, pages 463–470.

S. Chakrabarti. 2003. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann.

Z. Chen, S. Liu, L. Wenyin, Ge. Pu, and W. Ma. 2003. Building a web thesaurus from web link structure. In *Proc. of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 48–55.

K. Church and P. Hanks. 1991. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16:22–29.

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74.

R. Guha and A. Garg. 2004. Disambiguating people in search. In *Stanford University*.

T. Hisamitsu and Y. Niwa. 2001. Topic-word selection based on combinatorial probability. In *Proc. of NLPRS'01*, pages 289–296.

T. Hokama and H. Kitagawa. 2006. Extracting mnemonic names of people from the web. In *Proc. of 9th International Conference on Asian Digital Libraries (ICADL'06)*, pages 121–130.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proc. of the ACM conference on Knowledge Discovery and Data Minning (KDD)*.

T. Kudo, K. Yamamoto, and Y. Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proc. of EMNLP'04*.

W. Lu, L. Chien, and H. Lee. 2004. Anchor text mining for translation of web queries: A transitive translation approach. *ACM Transactions on Information Systems*, 22(2):242–269.

G. Salton and M.J. McGill. 1986. *Introduction to Modern Information Retreival*. McGraw-Hill Inc., New York, NY.

P. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proc. of the ACL*, pages 417–424.

Towards Automated Semantic Analysis on Biomedical Research Articles

Donghui Feng **Gully Burns** **Jingbo Zhu** **Eduard Hovy**
Information Sciences Institute
University of Southern California
Marina del Rey, CA, 90292
{donghui, burns, jingboz, hovy}@isi.edu

Abstract

In this paper, we present an empirical study on adapting Conditional Random Fields (CRF) models to conduct semantic analysis on biomedical articles using active learning. We explore uncertainty-based active learning with the CRF model to dynamically select the most informative training examples. This abridges the power of the supervised methods and expensive human annotation cost.

1 Introduction

Researchers have experienced an increasing need for automated/semi-automated knowledge acquisition from the research literature. This situation is especially serious in the biomedical domain where the number of individual facts that need to be memorized is very high.

Many successful information extraction (IE) systems, work in a supervised fashion, requiring human annotations for training. However, human annotations are either too expensive or not always available and this has become a bottleneck to developing supervised IE methods to new domains.

Fortunately, active learning systems design strategies to select the most informative training examples. This process can achieve certain levels of performance faster and reduce human annotation (e.g., Thompson et al., 1999; Shen et al., 2004).

In this paper, we present an empirical study on adapting CRF model to conduct semantic analysis on biomedical research literature. We integrate an uncertainty-based active learning framework with the CRF model to dynamically select the most informative training examples and reduce human annotation cost. A systematic study with exhaustive experimental evaluations shows that it can achieve satisfactory performance on biomedical data while requiring less human annotation.

Unlike direct estimation on target individuals in traditional active learning, we use two heuristic certainty scores, *peer comparison certainty* and *set comparison certainty*, to indirectly estimate sequences labeling quality in CRF models.

We partition biomedical research literature by experimental types. In this paper, our goal is to analyze various aspects of useful knowledge about tract-tracing experiments (TTE). This type of experiments has prompted the development of several curated databases but they have only partial coverage of the available literature (e.g., Stephan et al., 2001).

2 Related Work

Knowledge Base Management Systems allow individual users to construct personalized repositories of knowledge statements based on their own interaction with the research literature (Stephan et al., 2001; Burns and Cheng, 2006). But this process of data entry and curation is manual. Current approaches on biomedical text mining (e.g., Srinivas et al., 2005; OKanohara et al., 2006) tend to address the tasks of named entity recognition or relation extraction, and our goal is more complex: to extract computational representations of the minimum information in a given experiment type.

Pattern-based IE approaches employ seed data to learn useful patterns to pinpoint required fields values (e.g. Ravichandran and Hovy, 2002; Mann and Yarowsky, 2005; Feng et al., 2006). However, this only works if the data corpus is rich enough to learn variant surface patterns and does not necessarily generalize to more complex situations, such as our domain problem. Within biomedical articles, sentences tend to be long and the prose structure tends to be more complex than newsprint.

The CRF model (Lafferty et al., 2001) provides a compact way to integrate different types of features for sequential labeling problems. Reported work includes improved model variants (e.g., Jiao et al., 2006) and applications such as web data extraction (Pinto et al., 2003), scientific citation extraction (Peng and McCallum, 2004), word alignment (Blunsom and Cohn, 2006), and discourse-level chunking (Feng et al., 2007).

Pool-based active learning was first successfully applied to language processing on text classification (Lewis and Gale, 1994; McCallum and Nigam, 1998; Tong and Koller, 2000). It was also gradually applied to NLP tasks, such as information extraction (Thompson et al., 1999); semantic parsing (Thompson et al., 1999); statistical parsing (Tang et al., 2002); NER (Shen et al., 2004); and Word Sense Disambiguation (Chen et al., 2006). In this paper, we use CRF models to perform a more complex task on the primary TTE experimental results and adapt it to process new biomedical data.

3 Semantic Analysis with CRF Model

3.1 What knowledge is of interest?

The goal of TTE is to chart the interconnectivity of the brain by injecting tracer chemicals into a region of the brain and then identifying corresponding labeled regions where the tracer is transported to. A typical TTE paper may report experiments about one or many labeled regions.

Name	Description
injectionLocation	the named brain region where the injection was made.
tracerChemical	the tracer chemical used.
labelingLocation	the region/location where the labeling was found.
labelingDescription	a description of labeling, density or label type.

Table 1. Minimum knowledge schema for a TTE.

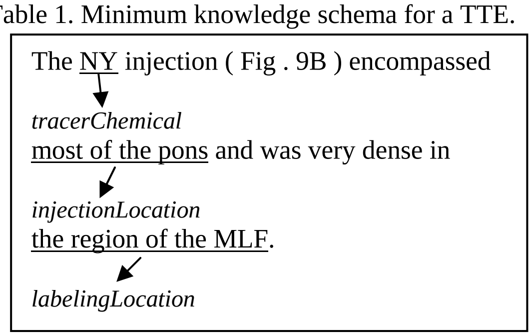


Figure 1. An extraction example of TTE description.

In order to construct the minimum information required to interpret a TTE, we consider a set of specific components as shown in Table 1.

Figure 1 gives an example of description of a complete TTE in a single sentence. In the research articles, this information is usually spread over many such sentences.

3.2 CRF Labeling

We use a plain text sentence for input and attempt to label each token with a field label. In addition to the four pre-defined fields, a default label, "O", is used to denote tokens beyond our concern.

In this task, we consider five types of features based on language analysis as shown in Table 2.

Name	Feature	Description
Lexical Knowledge	TOPOGRAPHY	Is word topographic?
	BRAIN_REGION	Is word a region name?
	TRACER	Is word a tracer chemical?
	DENSITY	Is word a density term?
	LABELING_TYPE	Does word denote a labeling type?
Surface Word	Word	Current word
Context Window	CONT_INJ	If current word if within a window of injection context
Window Words	Prev-word	Previous word
	Next-word	Next word
Dependency Features	Root-form	Root form of the word if different
	Gov-verb	The governing verb
	Subj	The sentence subject
	Obj	The sentence object

Table 2. The features for system labeling.

Lexical Knowledge. We define lexical items representing different aspects of prior knowledge. To this end we use names of brain structures taken from brain atlases, standard terms to denote neuroanatomical topographical spatial relationships, and common sense words for labeling descriptions. We collect five separate lexicons as shown in Table 3.

Lexicons	# of terms	# of words
BRAIN_REGION	1123	5536
DENSITY	8	10
LABELING_TYPE	9	13
TRACER	30	30
TOPOGRAPHY	9	36
Total	1179	5625

Table 3. The five lexicons.

Surface word. The word token is an important indicator of the probable label for itself.

Context Window. The TTE is a description of the inject-label-findings context. Whenever we find a word with a root form of "injection" or "deposit", we generate a context window around this word and all the words falling into this window are assigned a feature of "CON_INJ". This means when labeling these words the system should consider the very current context.

Window Words. We also use all the words occurring in the window around the current word. We set the window size to only include the previous and following words (window size = 1).

Dependency Features. To untangle word relationships within each sentence, we apply the dependency parser MiniPar (Lin, 1998) to parse each sentence, and then derive four types of features. These features are (a) root form of word, (b) the subject in the sentence, (c) the object in the sentence, and (d) the governing verb for each word.

4 Uncertainty-based Active Learning

Active learning was initially introduced for classification tasks. The intuition is to always add the most informative examples to the training set to improve the system as much as possible.

We apply an uncertainty/certainty score-based approach. Unlike traditional classification tasks, where disagreement or uncertainty is easy to obtain on target individuals, information extraction tasks in our problem take a whole sequence of tokens that might include several slots as processing units. We therefore need to make decisions on whether a full sequence should be returned for labeling.

Estimations on confidence for single segments in the CRF model have been proposed by (Culotta and McCallum, 2004; Kristjannson et al., 2004). However as every processing unit in the data set is at the sentence level and we make decisions at the sentence level to train better sequential labeling models, we define heuristic scores at the sentence level.

Symons et al. (2006) presents multi-criterion for active learning with CRF models, but our motivation is from a different perspective. The labeling result for every sentence corresponds to a decoding path in the state transition network. Inspired by the decoding and re-ranking approaches in statistical machine translation, we use two heuristic scores to measure the degree of correctness of the top labeling path, namely, *peer comparison certainty* and *set comparison certainty*.

Suppose a sentence *S* includes *n* words/tokens and a labeling path at position *m* in the ranked N-best list is represented by $L^m = (l_0, l_1, ..., l_{n-1})$. Then the probability of this labeling path is represented by $P(L^m)$, and we have the following two equations to define the peer comparison certainty score, $Score_{peer}(S)$ and set comparison certainty score, $Score_{set}(S)$:

$$Score_{peer}(S) = \frac{P(L^1)}{P(L^2)} \quad (1)$$

$$Score_{set}(S) = \frac{P(L^1)}{\sum_{k=1}^{N} P(L^k)} \quad (2)$$

For peer comparison certainty (Eq. 1), we calculate the ratio of the top-scoring labeling path probability to the second labeling path probability. A high ratio means there is a big jump from the top labeling path to the second one. The higher the ratio score, the higher the relative degree of correctness for the top labeling path, giving system higher confidence for those with higher peer comparison certainty scores. Sentences with lowest certainty score will be sent to the oracle for manual labeling.

In the labeling path space, if a labeling path is strong enough, its probability score should dominate all the other path scores. In Equation 2, we compute the set comparison certainty score by considering the portion of the probability of the path in the overall N-best labeling path space. A large value means the top path dominates all the other labeling paths together giving the system a higher confidence on the current path over others.

We start with a seed training set including *k* labeled sentences. We then train a CRF model with the training data and use it to label unlabeled data. The results are compared based on the certainty scores and those sentences with the lowest certainty scores are sent to an oracle for human labeling. The new labeled sentences are then added to the training set for next iteration.

5 Experimental Results

We first investigated how the active learning steps could help for the task. Second, we evaluated how the CRF labeling system worked with different sets of features. We finally applied the model to new

biomedical articles and examined its performance on one of its subsets.

5.1 Experimental Setup

We have obtained 9474 *Journal of Comparative Neurology (JCN)*[1] articles from 1982 to 2005. For sentence labeling, we collected 21 TTE articles from the JCN corpus. They were converted from PDF files to XML files, and all of the article sections were identified using a simple rule-based approach. As most of the meaningful descriptions of TTEs appear in the Results section, we only processed the Results section. The 21 files in total include 2009 sentences, in which 1029 sentences are meaningful descriptions for TTEs and 980 sentences are not related to TTEs.

We randomly split the sentences into a training pool and a testing pool, under a ratio 2:1. The training pool includes 1338 sentences, with 685 of them related to TTEs, while 653 not. Testing was based on meaningful sentences in the testing pool. Table 4 gives the configurations in the data pools.

	# of Related Sentences	# of Unrelated Sentences	Sum
Training Pool	685	653	1338
Testing Pool	344	327	671
Sum	1029	980	2009

Table 4. Training and testing pool configurations.

5.2 Evaluation Metrics

As the label "O" dominates the data set (70% out of all tokens), a simple accuracy score would provide an inappropriate high score for a baseline system that always chooses "O". We used Precision, Recall, and F_Score to evaluate only meaningful labels.

5.3 How well does active learning work?

For the active learning procedure, we initially selected a set of seed sentences related to TTEs from the training pool. At every step we trained a CRF model and labeled sentences in the rest of the training pool. As described in section 4, those with the lowest rank on certainty scores were selected. If they are related to a TTE, human annotation will be added to the training set. Otherwise, the system will keep on selecting sentences until it finds enough related sentences.

People have found active learning in batch mode is more efficient, as in some cases a single additional training example will not improve a classifier/system that much. In our task, we chose the bottom *k* related sentences with the lowest certainty scores. We conducted various experiments for *k* = 2, 5, and 10. We also compared experiments with passive learning, where at every step the new *k* related sentences were randomly selected from the corpus. Figures 2, 3, and 4 give the learning curves for precision, recall, and F_Scores when *k* = 10.

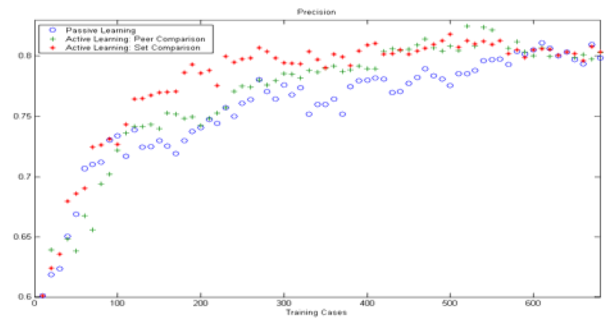


Figure 2. Learning curve for Precision.

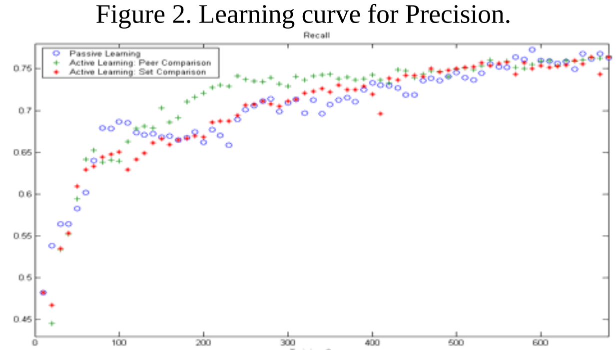


Figure 3. Learning curve for Recall.

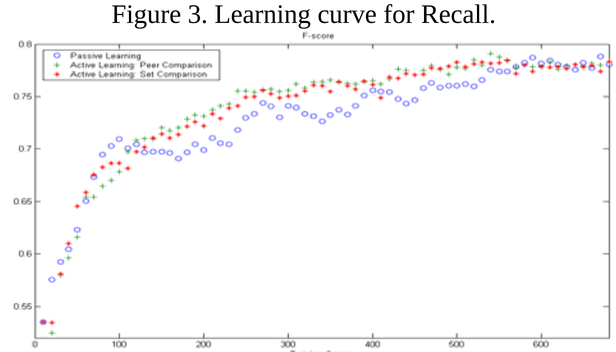


Figure 4. Learning curve for F_Score.

From these figures, we can see active learning approaches required fewer training examples to achieve the same level of performance. As we iteratively added new labeled sentences into the training set, the precision scores of active learning were steadily better than that of passive learning as the uncertain examples were added to strengthen

[1] http://www3.interscience.wiley.com/cgi-bin/jhome/31248

existing labels. However, the recall curve is slightly different. Before some point, the recall score of passive learning was a little better than active learning. The reason is that examples selected by active learning are mainly used to foster existing labels but have relatively weaker improvements for new labels, while passive learning has the freedom to add new knowledge for new labels and improve recall scores faster. As we keep on using more examples, the active learning catches up with and overtakes passive learning on recall score.

These experiments demonstrate that under the framework of active learning, examples needed to train a CRF model can be greatly reduced and therefore make it feasible to adapt to other domains.

5.4 How well does CRF labeling work?

As we added selected annotated sentences, the system performance kept improving. We investigated system performance at the final step when all the related sentences in the training pool are selected into the training set. The testing set also only includes the related sentences. This results in 685 training sentences and 344 testing sentences.

To establish a baseline for our labeling task, we simply scanned every sentence for words or phrases from each lexicon. If the term was present, then we labeled the word based on the lexicon in which it appeared. If words appeared in multiple lexicons, we assigned labels randomly.

System Features	Prec.	Recall	F_Score
Baseline	0.4067	0.1761	0.2458
Lexicon	0.5998	0.3734	0.4602
Lexicon + Surface Words	0.7663	0.7302	0.7478
Lexicon + Surface Words + Context Window	0.7717	0.7279	0.7491
Lexicon + Surface Words + Context Window + Window Words	0.8076	0.7451	0.7751
Lexicon + Surface Words + Context Window + Window Words + Dependency Features	**0.7991**	**0.7828**	**0.7909**

Table 5. Precision, Recall, and F_Score for labeling.

We tried exhaustive feature combinations. Table 5 shows system performance with different feature combinations. All systems performed significantly higher than the baseline. The sole use of lexicon knowledge produced poor performance, and the inclusion of surface words produced significant improvement. The use of window words boosted precision and recall. The performance with all the features generated an F_score of 0.7909.

We explored how system performance reflects different labels. Figure 5 and 6 depict the detailed distribution of system labeling from the perspective of precision and recall respectively for the system with the best performance. Most errors occurred in the confusion of *injectionLocation* and *labelingLocation*, or of the meaningful labels and "O".

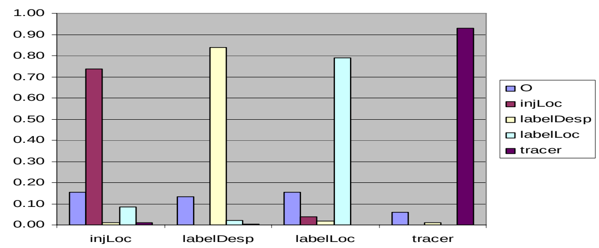


Figure 5. Precision confusion matrix distribution.

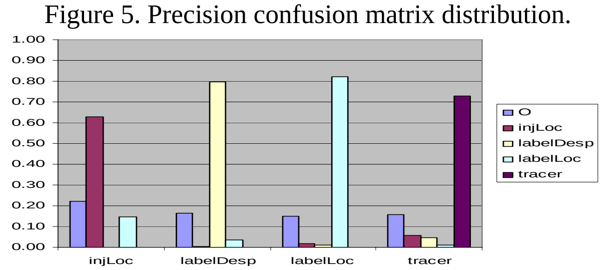


Figure 6. Recall confusion matrix distribution.

The worst performance occurred for files that distinguish themselves from others by using fairly different writing styles. We believe given more training data with different writing styles, the system could achieve a better overall performance.

5.5 On New Biomedical Data

Under this active learning framework, we have shown a CRF model can be trained with less annotation cost than using traditional passive learning. We adapted the trained CRF model to new biomedical research articles.

Out of the 9474 collected JCN articles, more than 230 research articles are on TTEs. The whole processing time for each document varies from 20 seconds to 90 seconds. We sent the new system-labeled files back to a biomedical knowledge expert for manual annotation. The time to correct one automatically labeled document is dramatically reduced, around 1/3 of that spent on raw text.

We processed 214 new research articles and examined a subset including 16 articles. We evalu-

ated it in two aspects: the overall performance and the performance averaged at the document level.

Table 6 gives the performance on the whole new subset and that averaged on 16 documents. The performance is a little bit lower than reported in the previous section as the new document set might include different styles of documents. We examined system performance at each document. Figure 7 gives the detailed evaluation for each of the 16 documents. The average F_Score of the document level is around 74%. For those documents with reasonable TTE description, the system can achieve an F_Score of 87%. The bad documents had a different description style and usually mixed the TTE descriptions with general discussion.

	Prec.	Recall	F_Score
Overall	0.7683	0.7155	0.7410
Averaged per Doc.	0.7686	0.7209	0.7418

Table 6. Performance on the whole new subset and the averaged performance per document.

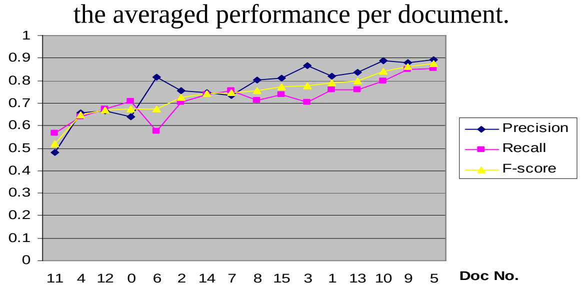


Figure 7. System performance per document.

6 Conclusions and Future Work

In this paper, we explored adapting a supervised CRF model for semantic analysis on biomedical articles using an active learning framework. It abridges the power of the supervised approach and expensive human costs. We are also investigating the use of other certainty measures, such as averaged field confidence scores over each sentence.

In the long run we wish to generalize the framework to be able to mine other types of experiments within the biomedical research literature and impact research in those domains.

References

Blunsom, P. and Cohn, T. 2006. Discriminative word alignment with conditional random fields. In *ACL-2006*.

Burns, G.A. and Cheng, W.C. 2006. Tools for knowledge acquisition within the NeuroScholar system and their application to anatomical tract-tracing data. In *Journal of Biomedical Discovery and Collaboration*.

Chen, J., Schein, A., Ungar, L., and Palmer, M. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proc. of HLT-NAACL 2006*.

Culotta, A. and McCallum, A. 2004. Confidence estimation for information extraction. In *HLT-NAACL-2004, short papers*.

Feng, D., Burns, G., and Hovy, E.H. 2007. Extracting data records from unstructured biomedical full text. In *Proc. of EMNLP-CONLL-2007*.

Feng, D., Ravichandran, D., and Hovy, E.H. 2006. Mining and re-ranking for answering biographical queries on the web. In *Proc. of AAAI-2006*.

Jiao, F., Wang, S., Lee, C., Greiner, R., and Schuurmans, D. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proc. of ACL-2006*.

Kristjannson, T., Culotta, A., Viola, P., and McCallum, A. 2004. Interactive information extraction with constrained conditional random fields. In *Proc. of AAAI-2004*.

Lafferty, J., McCallum, A., and Pereira, F. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML-2001*.

Lewis, D.D. and Gale, W.A. 1994. A sequential algorithm for training text classifiers. In *Proc. of SIGIR-1994*.

Lin, D. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.

Mann, G.S. and Yarowsky, D. 2005. Multi-field information extraction and cross-document fusion. In *Proc. of ACL-2005*.

McCallum, A.K. 2002. *MALLET: a machine Learning for language toolkit*. http://mallet.cs.umass.edu.

McCallum, A. and Nigam, K. 1998. Employing EM in pool-based active learning for text classification. In *Proc. of ICML-98*.

OKanohara, D., Miyao, Y., Tsuruoka, Y., and Tsujii, J. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition**.** In *ACL-2006*.

Peng, F. and McCallum, A. 2004. Accurate information extraction from research papers using conditional random fields. In *Proc. of HLT-NAACL-2004*.

Pinto, D., McCallum, A., Wei, X., and Croft, W.B. 2003. Table extraction using conditional random fields. In *SIGIR-2003*.

Ravichandran, D. and Hovy, E.H. 2002. Learning surface text patterns for a question answering system. In *ACL-2002*.

Shen, D., Zhang, J., Su, J., Zhou, G., and Tan, C.L. 2004. Multi-criteria-based active learning for named entity recognition. In *Proc. of ACL-2004*.

Srinivas, et al., 2005. Comparison of vector space model methodologies to reconcile cross-species neuroanatomical concepts. *Neuroinformatics*, 3(2).

Stephan, K.E., et al., 2001. Advanced database methodology for the Collation of Connectivity data on the Macaque brain (CoCoMac). *Philos Trans R Soc Lond B Biol Sci*.

Symons et al., 2006. Multi-Criterion Active Learning in Conditional Random Fields. In *ICTAI-2006*.

Tang, M., Luo, X., and Roukos, S. 2002. Active learning for statistical natural language parsing. In ACL-2002.

Thompson, C.A., Califf, M.E., and Mooney, R.J. 1999. Active learning for natural language parsing and information extraction. In *Proc. of ICML-99*.

Tong, S. and Koller, D. 2000. Support vector machine active learning with applications to text classification. In *Proc. of ICML-2000*.

Large Scale Diagnostic Code Classification for Medical Patient Records

Lucian Vlad Lita and **Shipeng Yu** and **Stefan Niculescu** and **Jinbo Bi**
Siemens Medical Solutions
firstname.lastname@siemens.com

Abstract

A critical, yet not very well studied problem in medical applications is the issue of accurately labeling patient records according to diagnoses and procedures that patients have undergone. This labeling problem, known as coding, consists of assigning standard medical codes (ICD9 and CPT) to patient records. Each patient record can have several corresponding labels/codes, many of which are correlated to specific diseases. The current, most frequent coding approach involves manual labeling, which requires considerable human effort and is cumbersome for large patient databases. In this paper we view medical coding as a multi-label classification problem, where we treat each code as a label for patient records. Due to government regulations concerning patient medical data, previous studies in automatic coding have been quite limited. In this paper, we compare two efficient algorithms for diagnosis coding on a large patient dataset.

1 Introduction

In order to be reimbursed for services provided to patients, hospitals need to provide proof of the procedures that they performed. Currently, this is achieved by assigning a set of CPT (Current Procedural Terminology) codes to each patient visit to the hospital. Providing these codes is not enough for receiving reimbursement: in addition, hospitals need to justify why the corresponding procedures have been performed. In order to do that, each patient visit needs to be coded with the appropriate diagnosis that require the above procedures. There are several standardized systems for patient diagnosis coding, with ICD9 (International Classification of Diseases, (Organization, 1997)) being the official version. Usually a CPT code is represented by a five digit integer whereas an ICD9 code is a real number consisting of a 2-3 digit disease category followed by 1-2 decimal subcategory. For example, a CPT code of 93307 is used for an Echo Exam. An ICD9 code of 428 represents Heart Failure (HF) with subcategories 428.0 (Congestive HF, Unspecified), 428.1 (Left HF), 428.2 (Systolic HF), 428.3 (Diastolic HF), 428.4(Combined HF) and 428.9 (HF, Unspecified).

The coding approach currently used in hospitals relies heavily on manual labeling performed by skilled and/or not so skilled personnel. This is a very time consuming process, where the person involved reads the patient chart and assigns the appropriate codes. Moreover, this approach is very error prone given the huge number of CPT and ICD9 codes. A recent study (Benesch et al., 1997) suggests that only 60%-80% of the assigned ICD9 codes reflect the exact patient medical diagnosis. This can be partly explained by the fact that coding is done by medical abstractors who often lack the medical expertise to properly reach a diagnosis. Two situations are prevalent: ”over-coding” (assigning a code for a more serious condition than it is justified) and ”under-coding” (missing codes for existing procedures/diagnoses). Both situations translate into significant financial loses: for insurance companies in the first case and for hospitals in the second case. Additionally, accurate coding is extremely important because ICD9 codes are widely used in determining patient eligibility for clinical trials as well as in quantifying hospital compliance with quality initiatives.

Another recent study (Sonel et al., 2006) stresses the importance of developing automated methods for patient record information extraction by demonstrating how an automated system performed with 8% better accuracy than a human abstractor on a task of identifying guideline compliance for unstable angina patients. In the study, differences between the automated system and the human abstractor were adjudi-

cated by an expert based on the evidence provided.

In this paper we compare several data mining techniques for automated ICD9 diagnosis coding. Our methods are able to predict ICD9 codes by modeling this task as a classification problem in the natural language processing framework. We demonstrate our algorithms in section 4 on a task of ICD9 coding of a large population of patients seen at a cardiac hospital.

2 Related Work

Classification under supervised learning setting has been a standard problem in machine learning or data mining area, which learns to construct inference models from data with known assignments, and then the models can be generalized to unseen data for code prediction. However, it has been rarely employed in the domain for automatic assignment of medical codes such as ICD9 codes to medical records. Part of the reason is that the data and labels are difficult to obtain. Hospitals are usually reluctant to share their patient data with research communities, and sensitive information (e.g. patient name, date of birth, home address, social security number) has to by anonymized to meet HIPAA (Health Insurance Portability and Accountability Act) (hip,) standards. Another reason is that the code classification task is itself very challenging. The patient records contain a lot of noise (misspellings, abbreviations, etc), and understanding the records correctly is very important to make correct code predictions.

Most of the ICD9 code assignment systems work with a rule-based engine as, for instance, the one available online from the site *http://www.icd9coding.com/*, or the one described in (reb,), which displays different ICD9 codes for a trained medical abstractor to look at and manually assign proper codes to patient records.

A health care organization can significantly improve its performance by implementing an automated system that integrates patients documents, tests with standard medical coding system and billing systems. Such a system offers large health care organizations a means to eliminate costly and inefficient manual processing of code assignments, thereby improving productivity and accuracy. Early efforts dedicated to automatic or semi-automatic assignments of ICD9 codes (Larkey and Croft, 1995; Lovis et al., 1995) demonstrate that simple machine learning approaches such as k-nearest neighbor, relevance feedback, or Bayesian independence classifiers can be used to acquire knowledge from already-coded training documents. The identified knowledge is then employed to optimize the means of selecting and ranking candidate codes for the test document. Often a combination of different classifiers produce better results than any single type of classifier. Occasionally, human interaction is still needed to enhance the code assignment accuracy (Lovis et al., 1995).

Similar work was performed to automatically categorize patients documents according to meaningful groups and not necessarily in terms of medical codes (de Lima et al., 1998; Ruch, 2003; Freitas-Junior et al., 2006; Ribeiro-Neto et al., 2001). For instance, in (de Lima et al., 1998), classifiers were designed and evaluated using a hierarchical learning approach. Recent works (Halasz et al., 2006) also utilize NGram techniques to automatically create Chief Complaints classifiers based on ICD9 groupings.

In (Rao et al.,), the authors present a small scale approach to assigning ICD9 codes of Diabetes and Acute Myocardial Infarction (AMI) on a small population of patients. Their approach is semi-automatic, consisting of association rules implemented by an expert, which are further combined in a probabilistic fashion. However, given the high degree of human interaction involved, their method will not be scalable to a large number of medical conditions. Moreover, the authors do not further classify the subtypes within Diabetes or AMI.

Very recently, the Computation Medicine Center was sponsoring an international challenge task on this type of text classification problem.[1] About $2,216$ documents are carefully extracted (including training and testing), and 45 ICD9 labels (with 94 distinct combinations) are used for these documents. More than 40 groups submitted their results, and the best macro and micro F1 measures are 0.89 and 0.77, respectively. The competition is a worthy effort in the sense that it provides a test bed to compare different algorithms. Unfortunately, public datasets are to date much smaller than the patient records in even a small hospital. Moreover, many of the documents are very simple (one or two sentences). It is difficult to train

[1] http://www.computationalmedicine.org/challenge/index.php

good classifiers based on such a small data set (even the most common label 786.2 (for "Cough") has only 155 reports to train on), and the generalizability of the obtained classifiers is also problematic.

3 Approach

This section describes the two data mining algorithms used in section 4 for assigning ICD9 codes to patient visits as well as the real world dataset used in our experiments.

3.1 Data: ICD-9 Codes & Patient Records

We built a 1.3GB corpus using medical patient records extracted from a real single-institution patient database. This is important since most published previous work was performed on very small datasets. Due to privacy concerns, since the database contains identified patient information, it cannot be made publicly available. Each document consists of a full hospital visit record for a particular patient. Each patient may have several hospital visits, some of which may not be documented if they choose to visit multiple hospitals[2]. Our dataset consists of 96557 patient visits, each of them being labeled with a one or more ICD9 codes. We have encountered 2618 distinct ICD9 codes associated with these visits, with the top five most frequent summarized in table 1. Given sufficient patient records supporting a code, this paper investigates the performance of statistical classification techniques. This paper focuses on correct classification of high-frequency diagnosis codes.

Automatic prediction of the ICD9 codes is a challenging problem. During each hospital visit, a patient might be subjected to several tests, have different lab results and undergo various treatments. For the majority of these events, physicians and nurses generate free text data either by typing the information themselves or by using a local or remote speech-to-text engine. The input method also affects text quality and therefore could impact the performance of classifiers based on this data. In addition to these obstacles for the ICD9 classification task, patient records often include medical history (i.e. past medical conditions, medications etc) and family history (i.e. parents' chronic diseases). By embedding unstructured medical information that does not directly describe a patient's state, the data becomes noisier.

A significant difference between medical patient record classification and general text classification (e.g. news domain) is word distribution. Depending on the type of institution, department profile, and patient cohort, phrases such as "discharge summary", "chest pain", and "ECG" may be ubiquitous in corpus and thus not carry a great deal of information for a classification task. Consider the phrase "chest pain": intuitively, it should correlate well with the ICD-9 code 786.50, which corresponds to the condition chest pain. However, through the nature of the corpus, this phrase appears in well over half of the documents, many of which do not belong to the 786.50 *category*.

3.2 Support Vector Machines

The first classification method consists of support vector machines (SVM), proven to perform well on textual data (Rogati and Yang, 2002). The experiments presented use the SVM Light toolkit (Joachims, 2002) with a linear kernel and a target positive-to-negative example ratio defined by the training data. We experiment with a cost function that assigns equal value to all classes, as well as with a target class cost equal to the ratio of negative to positive examples. The results shown in this paper correspond to SVM classifiers trained using the latter cost function. Note that better results may be obtained by tuning such parameters on a validation set.

3.3 Bayesian Ridge Regression

The second method we have tried on this problem is a probabilistic approach based on Gaussian processes. A Gaussian process (GP) is a stochastic process that defines a nonparametric prior over functions in Bayesian statistics (Rasmussen and Williams, 2006). In the linear case, i.e. the function has linear form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$, the GP prior on f is equivalent to a Gaussian prior on $\mathbf{w}$, which takes the form $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{w})$, with mean $\boldsymbol{\mu}_\mathbf{w}$ and covariance $\boldsymbol{\Sigma}_\mathbf{w}$. Then the likelihood of labels $\mathbf{y} = [y_1, \ldots, y_n]^\top$ is

$$P(\mathbf{y}) = \int \prod_{i=1}^{n} P(y_i|\mathbf{w}^\top \mathbf{x}_i) P(\mathbf{w}|\boldsymbol{\mu}_\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{w})\, d\mathbf{w} \quad (1)$$

with $P(y_i|\mathbf{w}^\top \mathbf{x}_i)$ the probability that document $\mathbf{x}_i$ takes label y_i.

[2]Currently, there is a movement to more portable electronic health records

ICD9	Freq	Coverage	Description
786.50	59957	0.621	Chest pain, unspecified
401.9	28232	0.292	Essential hypertension, unspecified
414.00	27872	0.289	Unspecified type of vessel, native or graft
427.31	15269	0.158	Atrial fibrillation
414.01	13107	0.136	Coronary atherosclerosis of native coronary artery

Table 1: Statistics of the top five ICD-9 codes most frequent in the patient record database. Frequency of ICD-9 code in corpus and the corresponding coverage (i.e. fraction of documents in the corpus that were coded with the particular ICD-9 code).

In general we fix $\boldsymbol{\mu}_{\mathbf{w}} = \mathbf{0}$, and $\boldsymbol{\Sigma}_{\mathbf{w}} = \mathbf{I}$ with $\mathbf{I}$ the identity matrix. Based on past experience we simply choose $P(y_i|\mathbf{w}^\top\mathbf{x}_i)$ to be a Gaussian, $y_i \sim \mathcal{N}(\mathbf{w}^\top\mathbf{x}_i, \sigma^2)$, with σ^2 a model parameter. Since everything is Gaussian here, the *a posteriori* distribution of $\mathbf{w}$ conditioned on the observed labels, $P(\mathbf{w}|\mathbf{y}, \sigma^2)$, is also a Gaussian, with mean

$$\hat{\boldsymbol{\mu}}_{\mathbf{w}} = \left(\mathbf{X}^\top\mathbf{X} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}, \qquad (2)$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ is a $n \times d$ matrix. The only model parameter σ^2 can also be optimized by maximizing the likelihood (1) with respect to σ^2. Finally for a test document $\mathbf{x}_*$, we predict its label as $\hat{\boldsymbol{\mu}}_{\mathbf{w}}^\top\mathbf{x}_*$ with the optimal σ^2. We can also estimate the variance of this prediction, but describing this is beyond the scope of this paper.

This model is sometimes called the *Bayesian ridge regression*, since the log-likelihood (i.e., the logarithm of (1)) is the negation of the ridge regression cost up to a constant factor (see, e.g., (Tikhonov and Arsenin, 1977; Bishop, 1995)):

$$\ell(\mathbf{y}, \mathbf{w}, \mathbf{X}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2,$$

with $\lambda = \sigma^2$. One advantage of Bayesian ridge regression is that there is a systematic way of optimizing λ from the data. Feature selection is done prior to calculation (2) to ensure the matrix inverse is feasible. Cholesky factorization is used to speed up calculation. Though the task here is classification, we treat the classification labels as regression labels and normalize them before learning (i.e., subtract the mean such that $\sum_i y_i = 0$).

4 Experiments

In this section we describe our experimental setup and results using the previously mentioned dataset and approaches. Each document in the patient database represents an event in the patient's hospital stay: e.g. radiology note, personal physician note, lab tests etc. These documents are combined to create a hospital visit profile and are subsequently preprocessed for the classification task. No stemming is performed for the experiments in this paper.

We limit our experiments on hospital visits with less than 200 doctor's notes. As a first pre-processing step, we eliminate redundancy at a paragraph level and we perform tokenization and sentence splitting. In addition, tokens go through a number and pronoun classing smoothing process, in which all numbers are replaced with the same token, and all person pronouns are replaced with a similar token. Further classing could be performed: e.g. dates, entity classing etc, but were not considered in these experiments. As a shared pre-processing for all classifiers, viable features are considered all unigrams with a frequency of occurrence greater or equal to 10 that do not appear in a standard lists of function words.

After removing consolidating patient visits from multiple documents, our corpus consists of near $100,000$ data points. We then randomly split the visits into training, validation, and test sets which contain 70%, 15%, and 15% of the corpus respectively. The classifiers were tested on an 15% unseen test set. Thus, the training set consists of approximately $57,000$ data points (patient visits), which is a more realistic dataset compared to the previously used datasets – e.g. the medical text dataset used in the Computation Medicine Center competition.

This paper presents experiments with the five most frequent ICD9 codes. This allows for more in-depth experiments with only a few labels and also ensures sufficient training and testing data for our experiments. From a machine learning perspective, most of the ICD9 codes are unbalanced: i.e. much less than half of the documents in the corpus actually have a given label. From a text processing perspective, this

	Average F1 Measure	
	Micro	**Macro**
SVM	0.683	0.652
BRR	0.688	0.667

Table 3: F1 measure for the ICD-9 classification experiments

is a normal multi-class classification setting.

Prior to training the classifiers on our dataset, we performed feature selection using χ^2. The top $1,500$ features with the highest χ^2 values were selected to make up the feature vector. The previous step in which the vocabulary was drastically reduced was necessary, since the χ^2 measure is unstable (Yang and Pedersen, 1997) when infrequent features are used. To generate the feature vectors, the χ^2 values were normalized into the ϕ coefficient and then each vector was normalized to an Euclidean norm of 1.

In these experiments, we have employed two classification approaches: support vector machine (SVM) and Bayesian ridge regression (BRR), for each of the ICD9 codes. We used the validation set to tune the specific parameters parameters for these approaches – all the final results are reported using the unseen test set. For the Bayesian ridge regression, the validation set is used to determine the λ parameter as well as the best cutting point for positive versus negative predictions in order to optimize the $F1$ measure. Training is very fast for both methods when $1,500$ features are selected using χ^2.

We evaluate our models using Precision, Recall, AUC (Area under the Curve) and F1 measure. The results on the top five codes for both classification approaches are shown in Table 2. For the same experiments, the receiver operating characteristic (ROC) curves of prediction are shown in Figure 1 and in Figure 2. The support vector machine and Bayesian ridge regression methods obtain comparable results on these independent ICD9 classification problems. The Bayesian ridge regression method obtains a slightly better performance.

It is important to note that the results presented in this section may considerably underestimate the true performance of our classifiers. Our classifiers are tested on ICD9 codes labeled by the medical abstractors, who, according to (Benesch et al., 1997), only have a 60%-80% accuracy. A better performance estimation can be obtained by adjudicating the differ-

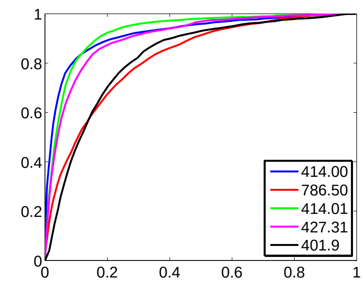


Figure 1: ROC curve for the SVM ICD9 classification

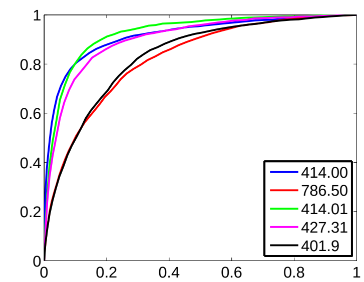


Figure 2: ROC curve for the BRR ICD9 classification

ences using a medical expert (as the small scale approach presented in (Sonel et al., 2006)), but we did not have access to such a resource.

5 Conclusions & Future Work

Code classification for medical patient records is becoming a critical task in the healthcare industry. This paper presents two automatic code classification approaches and applies them on a *r*eal, *l*arge hospital dataset. We view this problem as a multi-label classification problem and seek automatic solutions, specifically targeting ICD9 code classification. We have tested two state-of-the-art classification algorithms: support vector machines and Bayesian ridge regression) with promising performance.

The data set in our study contains more than 90,000 patient visits, and is by far the largest corpus for research purpose to the best of our knowledge. The features extracted from patient visits were selected for individual ICD9 codes based on χ^2 score. Low and high-frequency features were filtered out. Several other feature selection methods were considered (including information gain), yielding comparatively moderate performance levels.

ICD9	Support Vector Machine				Bayesian Ridge Regression			
	Prec	**Rec**	**F1**	**AUC**	**Prec**	**Rec**	**F1**	**AUC**
786.50	0.620	0.885	0.729	0.925	0.657	0.832	0.734	0.921
401.9	0.447	0.885	0.594	0.910	0.512	0.752	0.609	0.908
414.00	0.749	0.814	0.784	0.826	0.784	0.763	0.772	0.827
427.31	0.444	0.852	0.584	0.936	0.620	0.625	0.623	0.931
414.01	0.414	0.906	0.568	0.829	0.575	0.742	0.648	0.836

Table 2: Top five ICD-9 codes most frequent in the patient record database showing the performance of support vector machine-based method (SVM) and of bayesian ridge regression-based method (BRR).

Both Support Vector Machines and Bayesian ridge regression methods are fast to train and achieve comparable results. The F1 measure performance on the unseen test data is between 0.6 to 0.75 for the tested ICD9 codes, and the AUC scores are between 0.8 to 0.95. These results support the conclusion that automatic code classification is a promising research direction and offers the potential to change clinical coding dramatically.

Current approaches are still an incipient step towards more complex, flexible and robust coding models for classifying medical patient records. In current and future work we plan to employ more powerful models, extract more complex features, and explore inter-code correlations.

Patient record data exhibits strong correlations among certain ICD9 codes. For instance the code for fever 780.6 is very likely to co-occur with the code for cough 786.2. Currently we do not consider inter-code correlations and train separate classifier for individual codes. We are currently exploring methods that can take advantage of inter-code correlations and obtain a better, joint model for all ICD9 codes.

References

C. Benesch, D.M. Witter Jr, A.L. Wilder, P.W. Duncan, G.P. Samsa, and D.B. Matchar. 1997. Inaccuracy of the international classification of diseases (icd-9-cm) in identifying the diagnosis of ischemic cerebrovascular disease. *Neurology*.

C. M. Bishop. 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

Luciano R. S. de Lima, Alberto H. F. Laender, and Berthier A. Ribeiro-Neto. 1998. A hierarchical approach to the automatic categorization of medical documents. In *CIKM*.

H. R. Freitas-Junior, B. A. Ribeiro-Neto, R. De Freitas-Vale, A. H. F. Laender, and L. R. S. De Lima. 2006. Categorization-driven cross-language retrieval of medical information. *JASIST*.

S. Halasz, P. Brown, C. Goodall, D. G. Cochrane, and J. R. Allegra. 2006. The NGram cc classifier: A novel method of automatically creating cc classifiers based on ICD9 groupings. *Advances in Disease Surveillance*, 1(30).

Health insurance portability and accountability act. 2003. *http://www.hhs.gov/ocr/hipaa*.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines. Dissertation*. Kluwer.

L. Larkey and W. Croft. 1995. Automatic assignment of icd9 codes to discharge summaries.

Christian Lovis, P. A. Michel, Robert H. Baud, and Jean-Raoul Scherrer. 1995. Use of a conceptual semi-automatic icd-9 encoding system in a hospital environment. In *AIME*, pages 331–339.

World Health Organization. 1997. Manual of the international statistical classification or diseases, injuries, and causes of death. *World Health Organization, Geneva*.

R.B. Rao, S. Sandilya, R.S. Niculescu, C. Germond, and H. Rao. Clinical and financial outcomes analysis with existing hospital patient records. *SIGKDD*.

C. E. Rasmussen and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

PhyCor of Corsicana. In *Book Chapter of Information Technology for the Practicing Physician*. Springer London.

B. A. Ribeiro-Neto, A. H. F. Laender, and L. R. S. De-Lima. 2001. An experimental study in automatically categorizing medical documents. *JASIST*.

Monica Rogati and Yiming Yang. 2002. High-performing feature selection for text classification. *CIKM*.

P. Ruch. 2003. *Applying natural language processing to information retrieval in clinical records and biomedical texts*. Ph.D. thesis, Department of Informatics, Universite De Genéve.

A.F. Sonel, C.B. Good, H. Rao, A. Macioce, L.J. Wall, R.S. Niculescu, S. Sandilya, P. Giang, S. Krishnan, P. Aloni, and R.B. Rao. 2006. Use of remind artificial intelligence software for rapid assessment of adherence to disease specific management guidelines in acute coronary syndromes. *AHRQ*.

A. N. Tikhonov and V. Y. Arsenin. 1977. *Solutions of Ill-Posed Problems*. Wiley, New York.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. *ICML*.

Hacking Wikipedia for Hyponymy Relation Acquisition

Asuka Sumida **Kentaro Torisawa**
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi-shi, Ishikawa-ken, 923-1211 JAPAN
{a-sumida,torisawa}@jaist.ac.jp

Abstract

This paper describes a method for extracting a large set of hyponymy relations from Wikipedia. The Wikipedia is much more consistently structured than generic HTML documents, and we can extract a large number of hyponymy relations with simple methods. In this work, we managed to extract more than 1.4×10^6 hyponymy relations with 75.3% precision from the Japanese version of the Wikipedia. To the best of our knowledge, this is the largest machine-readable thesaurus for Japanese. The main contribution of this paper is a method for hyponymy acquisition from hierarchical layouts in Wikipedia. By using a machine learning technique and pattern matching, we were able to extract more than 6.3×10^5 relations from hierarchical layouts in the Japanese Wikipedia, and their precision was 76.4%. The remaining hyponymy relations were acquired by existing methods for extracting relations from definition sentences and category pages. This means that extraction from the hierarchical layouts almost doubled the number of relations extracted.

1 Introduction

The goal of this study has been to automatically extract a large set of hyponymy relations, which play a critical role in many NLP applications, such as Q&A systems (Fleischman et al., 2003). In this paper, hyponymy relation is defined as a relation between a hypernym and a hyponym when "the *hyponym* is a (kind of) *hypernym*."[1].

Currently, most useful sources of hyponymy relations are hand-crafted thesauri, such as WordNet (Fellbaum, 1998). Such thesauri are highly reliable, but their coverage is not large and the costs of extension and maintenance is prohibitively high. To reduce these costs, many methods have been proposed for automatically building thesauri (Hearst, 1992; Etzioni et al., 2005; Shinzato and Torisawa, 2004; Pantel and Pennacchiotti, 2006). But often these methods need a huge amount of documents and computational resources to obtain a reasonable number of hyponymy relations, and we still do not have a thesaurus with sufficient coverage.

In this paper, we attempt to extract a large number of hyponymy relations without a large document collection or great computational power. The key idea is to focus on Wikipedia[2], which is much more consistently organized than normal documents. Actually, some studies have already attempted to extract hyponymy relations or semantic classifications from Wikipedia. Hyponymy relations were extracted from definition sentences (Herbelot and Copestake, 2006; Kazama and Torisawa, 2007). Disambiguation of named entities was also attempted (Bunescu and Pasca, 2006). Category pages were used to extract semantic relations (Suchanek et al., 2007). Lexical patterns for semantic relations were learned (Ruiz-Casado et al., 2005).

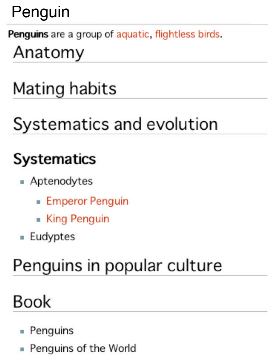
The difference between our work and these attempts is that we focus on the hierarchical layout of normal articles in Wikipedia. For instance, the article titled "Penguin" is shown in Fig. 1(b). This article has a quite consistently organized hierarchical structure. The whole article is divided into the sections "Anatomy", "Mating habits", "Systematics and evolution", "Penguins in popular culture" and so on. The section "Systematics and evolution" has the

[1]This is a slightly modified definition of the one in (Miller et al., 1990). Linguistic literature, e.g. (A.Cruse, 1998), distinguishes hyponymy relations, such as "national university" and "university", and concept-instance relations, such as "Tokyo University" and "university". However, we regard concept-instance relations as a part of hyponymy relations in this paper because we think the distinction is not crucial for many NLP applications.

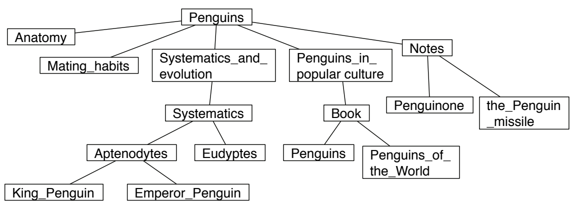
[2]http://ja.wikipedia.org/wiki

```
1: '''Penguins''' are a group of
   [[Aquatic animal|aquatic]],
   [[flightless bird]]s.
2: == Anatomy ==
3: == Mating habits ==
4: ==Systematics and evolution==
5: ===Systematics===
6: * Aptenodytes
7: **[[Emperor Penguin]]
8: ** [[King Penguin]]
9: * Eudyptes
10: == Penguins in popular culture ==
11: == Book ==
12: * Penguins
13: * Penguins of the World
14: == Notes ==
15: * Penguinone
16: * the [[Penguin missile]]
17: [[Category:Penguins]]
18: [[Category:Birds]]
```

(a) The source code of the article "Penguin' in 'Wikipedia



(b) The example of the article "Penguin" in Wikipedia



(c) The displayed page of the article "Penguin" in Wikipedia

Figure 1: The example of a Wikipedia article

subsection "Systematics", which is further divided to "Aptenodytes", "Eudyptes" and so on. Some of such section-subsection relations can be regarded as valid hyponymy relations. In the article about "Penguin", relations such as the one between "Aptenodytes" and "Emperor Penguin" and the one between "Book" and "Penguins of the World" are valid hyponymy relations. The main objective of this work is to develop a method to extract *only* such hyponymy relations.

The rest of the paper is organized as follows. We first explain the structure of Wikipedia in Section 2. Next, we introduce our method in Section 3. Some alternative methods are presented in Section 4. We then show the experimental results in Section 5.

2 The Structure of Wikipedia

The Wikipedia is built on the MediaWiki software package[3]. MediaWiki interprets the source code written in the MediaWiki syntax to produce human-readable web pages. For example, Fig. 1(b) is a result of interpreting the source code in Fig. 1(a). An important point is that the MediaWiki syntax is stricter than the HTML syntax and usage of the syntax in most Wikipedia articles are constrained by editorial policy. This makes it easier to extract information from the Wikipedia than from generic HTML documents.

[3]http://www.mediawiki.org/wiki/MediaWiki

Usually, a Wikipedia article starts with a definition sentence, such as "Penguins are a group of aquatic, flightless birds" in Fig. 1(a). Then, the hierarchical structure marked in the following manner follows.

Headings Headings describe the subject of a paragraph. See line 2-5, 10-11, 14 of Fig. 1(a). Headings are marked up as "=+*title*=+" in the MediaWiki syntax, where *title* is a subject of the paragraph. Note that "+" here means a finite number of repetition of symbols. "=+*section*=+" means that "=*section*=", "==*section*==" and "===*section*===" are legitimate mark up in the Wikipedia syntax. We use this '+' notation in the following explanation as well.

Bulleted lists Bulleted lists are lists of unordered items. See line 6-9, 12-13, 15-16 of Fig. 1. Bulleted lists are marked as "*+*title*" in the MediaWiki syntax, where *title* is a subject of a listed item.

Ordered lists Ordered lists are lists of numbered items. Ordered lists are marked up as "#+*title*" in MediaWiki syntax, where *title* is a subject of a numbered item.

Definition lists Definition lists contain terms and its definitions. Our method focuses only on the terms. Definition lists are marked as ";*title*" where *title* is a term.

The basic hierarchical structure of a Wikipedia article is organized by a pre-determined ordering among the above items. For instance, a bulleted list item is assumed to occupy a lower position in the hierarchy than a heading item. In general, items occupy a higher position in the order of headings, definition lists, bulleted lists, and ordered lists. In addition, recall that headings, bullet list and ordered list allowed the repetitions of symbols "=", "*" and "#". The number of repetition indicates the position in the hierarchy and the more repetition the item contains, the lower the position occupied by the item becomes. For instance, "==Systematics and evolution==" occupies a higher position than "===Systematics===" as illustrated in Fig. 1(a) (b).

Then, it is easy to extract a hierarchical structure based on the order among the mark-up items by parsing the source code of an article. Fig. 1(c) illustrates the hierarchical structure extracted from the source code in Fig. 1(a).

3 Proposed Method

This section describes our method for extracting hyponymy relations from hierarchical structures in Wikipedia articles. The method consists of three steps:

Step 1 Extract hyponymy relation candidates from hierarchical structures in the Wikipedia.

Step 2 Select proper hyponymy relations by applying simple patterns to the extracted candidates.

Step 3 Select proper hyponymy relations from the candidates by using a machine learning technique.

Each step is described below.

3.1 Step 1: Extracting Relation Candidates

The Step 1 procedure extracts the *title* of a marked-up item and a *title* of its (direct) subordinate marked-up item as a hyponymy relation for each marked-up item. For example, given the hierarchy in Fig. 1(c), the Step1 procedure extracted hyponymy relation candidates such as "Aptenodytes/Emperor Penguin"and "Book/Penguins of the World". (Note that we denote hyponymy relations or their candidates as "*hypernym*/*hyponym*" throughout this paper.) However, these relation candidates include many wrong hyponymy relations such as "Penguins in popular culture/Book". Steps 2 and 3 select proper relations from the output of the Step 1 procedure.

3.2 Step 2: Selecting Hyponymy Relations by Simple Patterns

Step 2 selects plausible hyponymy relations by applying simple patterns to hyponymy relation candidates obtained in Step 1. This is based on our observation that if a hypernym candidate matches a particular pattern, it is likely to constitute a correct relation. For example, in Japanese, if a hypernym candidate is " `omona` X (Popular *or* typical X)", X is likely to be a correct hypernym of the hyponym candidates that followed it in the article. Fig.2 shows a Japanese Wikipedia article about a zoo that includes "`omona doubutsu` (Popular animals)", " `Mazeran Pengin` (Magellanic Penguin)", "`Raion` (Lion)" and so on. From this article, the Step 1 procedure extracts a hyponymy relation candidate "Popular Animals/Magellanic Penguin", and the Step 2 procedure extracts "Animals/Magellanic Penguin" after matching "Popular" to the hypernym candidate and removing the string "Popular" from the candidate. Fig. 3 lists all the patterns we used. Note that the non-variable part of the patterns is removed from the matched hypernym candidates.



Figure 2: Example for Step2

`Xno ichiran`(list of X), `Xichiran`(list of X), `Xsyousai`(details of X), `Xrisuto`(X list), `daihyoutekinaX`(typical X), `daihyouX`(typical X), `syuyounaX`(popular or typical X), `omonaX`(popular or typical X), `syuyouX`(popular or typical X), `kihontekinaX`(basic X), `kihon`(basic X), `chomeinaX`(notable X), `ookinaX`(large X), `omonaX`(popular or typical X), `ta noX`(other X), `ichibuX`(partial list of X)

Figure 3: Patterns for Step 2

3.3 Step 3: Selecting Proper Hyponymy Relations by Machine Learning

The Step 3 procedure selects proper hyponymy relations from the relation candidates that do not match the patterns in Step 2. We use Support Vector Machines (SVM) (Vapnik, 1998) for this task. For each hyponymy relation candidate, we firstly apply morphological analysis and obtain the following types of features for each hypernym candidate and hyponym candidate, and append them into a single feature vector, which is given to the classifier.

POS We found that POS tags are useful clues for judging the validity of relations. For instance, if a hypernym includes proper nouns (and particularly toponyms), it is unlikely to constitute a proper relation. We assigned each POS tag a unique dimension in the feature space and if a hypernym/hyponym consists of a morpheme with a particular POS tag, then the corresponding element of the feature vector was set to one. When hypernyms/hyponyms are multiple morpheme expressions, the feature vectors for every morpheme were simply summed. (The obtained feature vector works as *disjunction* of each feature vector.) An important point is that, since the last morpheme of hypernyms/hyponyms works as strong evidence for the validity of relations, the POS tag of the last morpheme was mapped to the dimension that is different from the POS tags of the other morphemes.

MORPH Morphemes themselves are also mapped to a dimension of the feature vectors. The last morphemes are also mapped to dimensions that are different from those of the other morphemes. This feature is used for recognizing particular morphemes that strongly suggest the validity of hyponymy relations. For instance, if the morpheme "`zoku` (genus)" comes in the end of the hypernym, the relation is likely to be valid, as exemplified by the relation "`koutei pengin zoku` (Aptenodytes genus)/`koutei pengin` (Emperor Penguin)".

EXP Expressions of hypernym/hyponym candidates themselves also give a good clue for judging the validity of the relation. For instance, there are typical strings that can be the title of a marked-up item but cannot be a proper hypernym or a proper hyponym. Examples of these strings include "Background" and "Note". By mapping each expression to an element in a feature vector and setting the element to one, we can prevent the candidates containing such expressions from being selected by the classifier.

ATTR We used this type of features according to our observation that if a relation candidate includes an *attribute*, it is a wrong relation. The attributes of an object can be defined as "what we want to know about the object". For instance, we regard "Anatomy" as attributes of creatures in general, and the relation such as "Penguin/Anatomy" cannot be regarded as proper hyponymy relations. To set up this type of features, we automatically created a set of attributes and the feature was set to one if the hypernym/hyponym is included in the set. The attribute set was created in the following manner. We collected all the titles of the marked-up items from all the articles, and counted the occurrences of each title. If a title appears more than one time, then it was added to the attribute set. Note that this method relies on the hypothesis that the same attribute is used in articles about more than one object (e.g., "Penguin" and "Sparrow") belonging to the same class (e.g., "animal"). (Actually, in this counting of titles, we excluded the titles of items in the bulleted lists and the ordered lists in the bottom layer of the hierarchical structures. This is because these items are likely to constitute valid hyponymy relations. We also excluded that match the patterns in Fig. 3.) As a result, we obtained the set of 40,733 attributes and the precision of a set was 73% according to the characterization of attributes in (Tokunaga et al., 2005).

LAYER We found that if a hyponymy relation is extracted from the bottom of the hierarchy, it tends to be a correct relation. For example, in Fig. 1(c), the hyponymy relation "Penguin/Anatomy" which is extracted from the top of hierarchy is wrong, but the hyponymy relation "Aptenodytes/Emperor Penguin " which is extracted from the bottom of the layer is correct. To capture this tendency, we added the mark that marks up a hypernym and a hyponym to the features. Each mark is mapped to a dimension in the feature vector, and the corresponding element was set to one if a hypernym/hyponym candidate appears with the mark.

As the final output of our method, we merged the results of Steps 2 and 3.

4 Alternative Methods

This section describes existing methods for acquiring hyponymy relations from the Wikipedia. We compare the results of these methods with the output of our method in the next section.

4.1 Extraction from Definition Sentences

Definition sentences in the Wikipedia article were used for acquiring hyponymy relations by (Kazama and Torisawa, 2007) for named entity recognition. Their method is developed for the English version of the Wikipedia and required some modifications to the Japanese version. These modification was inspired by Tsurumaru's method (Tsurumaru et al., 1986).

Basically, definition sentences have forms similar to "*hyponym word* wa *hypernym word* no isshu de aru(*hyponym* is a kind of *hypernym*)" in dictionaries in general, and contain hyponymy relations in them. In the Wikipedia, such sentences usually come just after the titles of articles, so it is quite easy to recognize them. To extract hyponymy relations from definition sentences, we manually prepared 1,334 patterns, which are exemplified in Table 4, and applied them to the first sentence.

4.2 Extraction from Category Pages

Suchanek et al. (Suchanek et al., 2007) extracted hyponymy relations from the category pages in the Wikipedia using WordNet information. Although we cannot use WordNet because there is no Japanese version of WordNet, we can apply their idea to the Wikipedia only.

The basic idea is to regard the pairs of the category name provided in the top of a category page and the

hyponym wa.**hypernym* no hitotsu.
(*hyponym* is one of *hypernym*)
hyponym wa .**hypernym* no daihyoutekina mono dearu.
(*hyponym* is a typical *hypernym*)
hyponym wa.**hypernym* no uchi no hitotsu.
(*hyponym* is one of *hypernym*)

Note that *hyponym* and *hypernym* match only with NPs.

Figure 4: Examples of patterns for definition sentences

items listed in the page as hyponymy relation.

Thus, the method is quite simple. But the relations extracted by this are not limited to hyponymy relations, unfortunately. For instance, the category page "football" includes "football team". Such loosely associated relations are harmful for obtaining precise relations. Suchanek used WordNet to prevent such relations from being included in the output. However, we could not develop such a method because of the lack of a Japanese WordNet.

5 Experiments

For evaluating our method, we used the Japanese version of Wikipedia from March 2007, which includes 820,074 pages[4]. Then, we removed "user pages","special pages", "template pages", "redirection pages", and "category pages" from it.

In Step 3, we used TinySVM[5] with polynomial kernel of degree 2 as a classifier. From the relation candidates given to the Step 3 procedure, we randomly picked up 2,000 relations as a training set, and 1,000 relations as a development set. We also used the morphological analyzer MeCab [6] in Step 3.

Table 1 summarizes the performance of our method. Each row of the table shows A) the precision of the hyponymy relations, B) the number of the relations, and C) the expected number of correct relations estimated from the precision and the number of the extracted relations, after each step of the procedure. Note that Step 2' indicates the hyponymy relation candidates that *did not* match the pattern in Fig.3 and that were given to the Step 3 procedure. The difference between Step 2' and Step 3 indicates the effect of our classifier. Step 2&3 is the final result obtained by merging the results of Step 2 and Step 3. As the final output, we obtained more than 6.3×10^5

[4]This pages include "incomplete pages" that are not counted in the number of pages presented in the top page of the Wikipedia.

[5]http://chasen.org/ taku/software/TinySVM/index.html

[6]http://mecab.sourceforge.net

Table 1: Performance of each step

	Precision	# of rels.	estimated # of correct rels.
Step 1	44%	2,768,856	1,218,296
Step 2	71.5%	221,605	158,447
Step 2'	40.0%	2,557,872	1,023,148
Step 3	78.1%	416,858	325,670
Step 2 & 3	76.4%	633,122	484,117

aatisuto / erubisu puresurii
Artist / Elvis Presley
sakura / someiyoshino
Cherry Blossom / Yoshino Cherry
heiya / nakagawa heiya
Plain / Nakagawa Plain
ikou oyobi kenzoubutsu / tsuki no piramiddo
Ruins and buildings / the Pyramid of the Moon
suponsaa / genzai*
Sponsors / Present*
shutsuen sakuhin / taidan go*
Art work / After leaving a group*

"*" indicates an incorrectly recognized relation.

Figure 5: Examples of acquired hyponymy relations

relations and their precision was 76.4%. Note that the precision was measured by checking 200 random samples for each step except for Step 3 and Step 2&3, for which the precision was obtained in a way described later. Note that all the numbers were obtained after removing duplicates in the relations. Example of the relations recognized by Step 2 or Step 3 are shown in Fig. 5.

Table 2 shows the effect of each type of features in Step 3. Each row indicates the precision, recall and F-measure against 400 samples that are randomly selected from the relation candidates given to Step 3, when we removed a type of features from feature vector and when we used all the types. (The 400 samples included 142 valid relations.) We can see that all types except for LAYER contributed to an improvement of the F-measure. When the LAYER features were removed, the F-measure was improved to 1.1 but the precision was on an unacceptable level (55%) and cannot be used in actual acquisition.

Table 3 summarizes the statistics of all the methods for acquisition from Wikipedia. It shows A) the pre-

Table 2: Effect of each features in Step3

Feature Type	a Precision	Recall	F-measure
-POS	60.0%	57.0%	58.4
-MORPH	85.0%	47.8%	61.2
-EXP	82.2%	35.9%	50.0
-ATTR	79.7%	47.1%	59.2
-LAYER	55.0%	76.7%	64.1
ALL	78.1%	52.8%	63.0

Table 3: The result for extracting hyponymy relations from definition sentences, category structures,and hierarchy structures

	Precision	# of rels.	# of correct rels.
Hierarchy (Proposed)	76.4 %	633,122	484,117
Definition snts	77.5%	220,892	171,191
Category	70.5%	596,463	420,506
Total	75.3%	1,426,861	1,075,814

cision of the relations (200 random samples), B) the number of relations, and C) the expected number of correct relations estimated from the precision and the number of extracted relations. We obtained 1.4×10^6 hyponymy relations without duplication in total with 75.3% precision from definition sentences, category structures, and hierarchical structures. They covered 6.6×10^5 distinct hyponyms and 1.0×10^5 distinct hypernyms. Note that the number of duplicated relations in these results was just 23,616. This suggests that we could extract different types of hyponymy relations from each of these methods.

6 Conclusion

This paper described a method for extracting a large set of hyponymy relations from the hierarchical structures of articles in Wikipedia. We could extract 633,122 relations from hierarchical layouts in the Japanese Wikipedia and their precision was 76.4%. Combining with existing methods that extract relations from definition sentences and category structures, we were able to extract 1,426,861 relations with 75.3% precision in total without duplication. To the best of our knowledge, this is the largest machine-readable thesaurus for Japanese available.

References

D. A.Cruse. 1998. *Lexical Semantics*. Cambridge Textbooks in Linguistics.

Razvan C. Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the EACL*, pages 9–16.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165(1):91–134.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. 2003. Offline strategies for online question answering: Answering questions before they are asked. In *ACL2003*, pages 1–7.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545.

Aurelie Herbelot and Ann Copestake. 2006. Acquiring ontological relationships from wikipedia using rmrs. In *Proceedings of the ISWC 2006 Workshop on Web Content Mining with Human Language Technologies*.

Jun'ichi Kazama and Kentaro Torisawa. 2007. Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to wordnet: An on-line lexical database. In *Journal of Lexicography*, pages 235–244.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *ACL '06 : Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 113–120.

Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. 2005. Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In *NLDB*, pages 67–79.

Keiji Shinzato and Kentaro Torisawa. 2004. Acquiring hyponymy relations from web documents. In *HLT-NAACL '04 : Proceedings of Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting*, pages 73–80.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *WWW '07 : Proceedings of the 16th International World Wide Web Conference*.

Kosuke Tokunaga, Jun'ichi Kazama, and Kentaro Torisawa. 2005. Automatic discovery of attribute words from web documents. In *IJCNLP 2005*, pages 106–118.

Hiroaki Tsurumaru, Toru Hitaka, and Sho Yoshida. 1986. An attempt to automatic thesaurus construction from an ordinary japanese language dictionary. In *Proceedings of the 11th conference on Computational linguistics*, pages 445–447.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

A Discriminative Approach to Japanese Abbreviation Extraction

Naoaki Okazaki†
okazaki@is.s.u-tokyo.ac.jp

Mitsuru Ishizuka†
ishizuka@i.u-tokyo.ac.jp

Jun'ichi Tsujii†‡
tsujii@is.s.u-tokyo.ac.jp

†Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

‡School of Computer Science, University of Manchester
National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre,
131 Princess Street, Manchester M1 7DN, UK

Abstract

This paper addresses the difficulties in recognizing Japanese abbreviations through the use of previous approaches, examining actual usages of parenthetical expressions in newspaper articles. In order to bridge the gap between Japanese abbreviations and their full forms, we present a discriminative approach to abbreviation recognition. More specifically, we formalize the abbreviation recognition task as a binary classification problem in which a classifier determines a positive (abbreviation) or negative (non-abbreviation) class, given a candidate of abbreviation definition. The proposed method achieved 95.7% accuracy, 90.0% precision, and 87.6% recall on the evaluation corpus containing 7,887 (1,430 abbreviations and 6,457 non-abbreviation) instances of parenthetical expressions.

1 Introduction

Human languages are rich enough to be able to express the same meaning through different diction; we may produce different sentences to convey the same information by choosing alternative words or syntactic structures. Lexical resources such as WordNet (Miller et al., 1990) enhance various NLP applications by recognizing a set of expressions referring to the same entity/concept. For example, text retrieval systems can associate a query with alternative words to find documents where the query is not obviously stated.

Abbreviations are among a highly productive type of term variants, which substitutes fully expanded terms with shortened term-forms. Most previous studies aimed at establishing associations between abbreviations and their full forms in English (Park and Byrd, 2001; Pakhomov, 2002; Schwartz and Hearst, 2003; Adar, 2004; Nadeau and Turney, 2005; Chang and Schütze, 2006; Okazaki and Ananiadou, 2006). Although researchers have proposed various approaches to solving abbreviation recognition through methods such as deterministic algorithm, scoring function, and machine learning, these studies rely on the phenomenon specific to English abbreviations: all letters in an abbreviation appear in its full form.

However, abbreviation phenomena are heavily dependent on languages. For example, the term *one-segment broadcasting* is usually abbreviated as *one-seg* in Japanese; English speakers may find this peculiar as the term is likely to be abbreviated as *1SB* or *OSB* in English. We show that letters do not provide useful clues for recognizing Japanese abbreviations in Section 2. Elaborating on the complexity of the generative processes for Japanese abbreviations, Section 3 presents a supervised learning approach to Japanese abbreviations. We then evaluate the proposed method on a test corpus from newspaper articles in Section 4 and conclude this paper.

2 Japanese Abbreviation Survey

Researchers have proposed several approaches to abbreviation recognition for non-alphabetical languages. Hisamitsu and Niwa (2001) compared different statistical measures (e.g., χ^2 test, log like-

Type	Para	#	(%)	Examples	
Acronym	o	90	(1.2)	東京大学（東大） Tokyo Daigaku (ToDai) The University of Tokyo (UoT)	首都圏中央連絡自動車道（圏央道） Shutoken Chuou Renraku Jidousha Dou (Ken-o-dou) Metropolitan Inter-City Expressway (Ken-o Exp)
Acronym with translation	o	717	(9.1)	夜間離着陸訓練（ＮＬＰ） Yakan Richakuriku Kunren (NLP) Night Landing Practice (NLP)	ワールドカップ（Ｗ杯） Warudo Kappu (W hai) World Cup (WC)
Alias	o	623	(7.9)	朝鮮民主主義人民共和国（北朝鮮） Cho-sen Minshushugi Jinmin Kyowakoku (Kita Chosen) Democratic People's Republic of Korea (North Korea)	2000年問題（Ｙ２Ｋ） 2000 Nen Mondai (Y2K) Year 2000 problem (Y2K)
Attribute (reading)	x			毅然（きぜん） Kizen (kizen) firm [fə:rm]	Ｏ（オー）１５７ O (O-) 157
Attribute (location)	x			つくば学園都市（茨城県つくば市） Tsukuba Gakuen Toshi (Ibaraki-ken Tsukuba-shi) Tsukuba Science City (Tukuba City, Ibaraki Pref.)	
Attribute (affiliation)	x	6,457	(81.9)	インディペンデント（英国） Indipendento (Eikoku) Independent (UK)	ミヒャエル・シューマッハー（独） Mihyaeru Shumahha- (GER) Michael Schumacher (GER)
Epexegesis	x			西ドイツ（当時） Nishi Doitsu (touji) West Germany (at that time)	平成金融再生機構（仮称） Heisei Kinyu Saisei Kikou (Kasho) Financial Revitalization Corporation (tentative name)
Others	x			…（中略） ... (churyaku) ... (snip)	

Table 1: Parenthetical expressions used in Japanese newspaper articles

lihood ratio) to assess the co-occurrence strength between the inner and outer phrases of parenthetical expressions *X (Y)*. Yamamoto (2002) utilized the similarity of local contexts to measure the paraphrase likelihood of two expressions based on the distributional hypothesis (Harris, 1954). Chang and Teng (2006) formalized the generative processes of Chinese abbreviations with a noisy channel model. Sasano et al. (2007) designed rules about letter types and occurrence frequency to collect lexical paraphrases used for coreference resolution.

How are these approaches effective in recognizing Japanese abbreviation definitions? As a preliminary study, we examined abbreviations described in parenthetical expressions in Japanese newspaper articles. We used the 7,887 parenthetical expressions that occurred more than eight times in Japanese articles published by the *Mainichi Newspapers* and *Yomiuri Shimbun* in 1998–1999. Table 1 summarizes the usages of parenthetical expressions in four groups. The field 'para' indicates whether the inner and outer elements of parenthetical expressions are interchangeable.

The first group *acronym* (I) reduces a full form to a shorter form by removing letters. In general, the process of acronym generation is easily interpreted: the left example in Table 1 consists of two Kanji letters taken from the heads of the two words, while the right example consists of the letters at the end of the 1st, 2nd, and 4th words in the full form. Since all letters in an acronym appear in its full form, previous approaches to English abbreviations are also applicable to Japanese acronyms. Unfortunately, in this survey the number of such 'authentic' acronyms amount to as few as 90 (1.2%).

The second group *acronym with translation* (II) is characteristic of non-English languages. Full forms are imported from foreign terms (usually in English), but inherit the foreign abbreviations. The third group *alias* (III) presents generic paraphrases that cannot be interpreted as abbreviations. For example, *Democratic People's Republic of Korea* is known as its alias *North Korea*. Even though the formal name does not refer to the 'northern' part, the alias consists of *Korea*, and the locational modifier *North*. Although the second and third groups retain their interchangeability, computers cannot recognize abbreviations with their full forms based on letters.

The last group (IV) does not introduce interchangeable expressions, but presents additional information for outer phrases. For example, a location usage of a parenthetical expression *X (Y)* describes an entity *X*, followed by its location *Y*. Inner and outer elements of parenthetical expressions are not interchangeable. We regret to find that as many as 81.9% of parenthetical expressions were described for this usage. Thus, this study regards acronyms (with and without translation) and alias as *Japanese*

#	Expression *Y*	Expression *X*	Freq	Class
1	北朝鮮 (North Korea)	朝鮮民主主義人民共和国 (Democratic People's Republic of Korea)	4160	A
2	W杯 (W Cup)	ワールドカップ (World Cup)	2891	T
3	EU	欧州連合 (European Union)	2638	T
4	NATO	北大西洋条約機構 (North Atlantic Treaty Organization)	2593	T
5	IMF	国際通貨基金 (International Monetary Fund)	2473	T
6	中国 (China)	人民日報 (People's Daily)	1561	O
7	IOC	国際オリンピック委員会 (International Olympic Committee)	1550	T
8	WTO	世界貿易機関 (World Trade Organization)	1504	T
9	独 (Germany)	ディ・ウェルト (Die Welt)	1484	O
10	エジプト (Egypt)	アルアハラム (Al-Ahram)	1350	O

Table 2: Top 10 frequent parenthetical expressions used in Japanese newspapers from 1998–1999

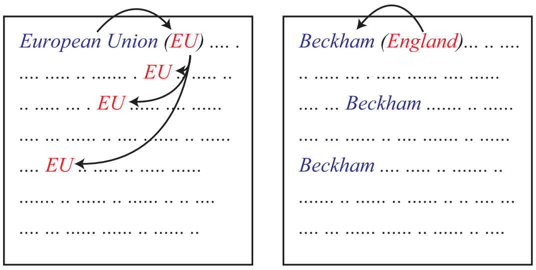
abbreviations in a broad sense, based on their interchangeabilities. In other words, the goal of this study is to classify parenthetical expressions *X (Y)* into true abbreviations (groups I, II, III) and other usages of parentheses (group IV).

How much potential do statistical approaches have to identify Japanese abbreviations? Table 2 shows the top 10 most frequently appearing parenthetical expressions in this survey. The 'class' field represents the category[1]: *T: acronym with translation*, *A: alias*, and *O: non-abbreviation*. The most frequently occurring parenthetical expression was *Democratic People's Republic of Korea (North Korea)* (4,160 occurrences). 7 instances in the table were acronyms with translation (#2–5, #7–8), and an alias (#1), but 3 non-abbreviation instances (#6, #9, and #10) expressed nationalities of information sources. Even if we designed a simple method to choose the top 10 parenthetical expressions, the recognition performance would be no greater than 70% precision.

3 A discriminative approach to abbreviation recognition

In order to bridge the gap between Japanese abbreviations and their full forms, we present a discriminative approach to abbreviation recognition. More specifically, we formalize the abbreviation recognition task as a binary classification problem in which

[1]No *acronym* was included in the top 10 list.



(a) Paraphrase (b) Non paraphrase

Figure 1: Paraphrase occurrence with parentheses

a classifier determines a positive (abbreviation) or negative (non-abbreviation) class, given a parenthetical expression *X (Y)*. We model the classifier by using Support Vector Machines (SVMs) (Vapnik, 1998). The classifier combines features that characterize various aspects of abbreviation definitions. Table 3 shows the features and their values for the abbreviation *EU*, and its full form: *O-shu Rengo (European Union)*. A string feature is converted into a set of boolean features, each of which indicates 'true' or 'false' of the value. Due to the space limitation, the rest of this section elaborates on *paraphrase ratio* and *SKEW* features.

Paraphrase ratio Let us consider the situation in which an author describes an abbreviation definition *X (Y)* to state a paraphrase $X \rightarrow Y$ in a document. The effect of the statement is to define the meaning of the abbreviation *Y* as *X* in case the reader may be unaware/uncertain of the abbreviation *Y*. For example, if an author wrote a parenthetical expression, *Multi-Document Summarization (MDS)*, in a document, readers would recognize the meaning of the expression *MDS*. Even if they were aware of the definition, *MDS* alone would be ambiguous; it could stand for *Multi Dimensional Scaling*, *Missile Defense System*, etc. Therefore, an author rarely uses the expression *Y* before describing its definition.

At the same time, the author would use the expression *Y* more than *X* after describing the definition, if it were to declare the abbreviation *Y* for *X*. Figure 1 illustrates this situation with two documents. Document (a) introduces the abbreviation *EU* for *European Union* because the expression *EU* occurs more frequently than *European Union* after the parenthetical expression. In contrast, the parenthetical expres-

Feature	Type	Description	Example
$\mathrm{PR}(X, Y)$	numeric	Paraphrase ratio	0.426
$\mathrm{SKEW}(X, Y)$	numeric	Similarity of local contexts measured by the skew divergence	1.35
$\mathrm{freq}(X)$	numeric	Frequency of occurrence of X	2,638
$\mathrm{freq}(Y)$	numeric	Frequency of occurrence of Y	8,326
$\mathrm{freq}(X, Y)$	numeric	Frequency of co-occurrence of X and Y	3,121
$\chi^2(X, Y)$	numeric	Co-occurrence strength measured by the χ^2 test	2,484,521
$\mathrm{LLR}(X, Y)$	numeric	Co-occurrence strength measured by the log-likelihood ratio	6.8
$\mathrm{match}(X, Y)$	boolean	Predicate to test whether X contains all letters in Y	0
Letter types	string	Pair of letter types of X and Y	Kanji/Alpha
First letter	string	The first letter in the abbreviation Y	*E*
Last letter	string	The last letter in the abbreviation Y	*U*
POS tags	string	Pair of POS tags for X and Y	NNP/NNP
POS categories	string	Pair of POS categories for X and Y	NN/NN
NE tags	string	Pair of NE tags for X and Y	ORG/ORG

Table 3: Features for the SVM classifier and their values for the abbreviation *EU*.

sion in document (b) describes the property (nationality) of a person *Beckham*.

Suppose that we have a document that has a parenthetical expression with expressions X and Y. We regard a document introducing an abbreviation Y for X if the document satisfies both of these conditions:

1. The expression Y appears more frequently than the expression X does after the definition pattern.

2. The expression Y does not appear before the definition pattern.

Formula 1 assesses the paraphrase ratio of the expressions X and Y,

$$\mathrm{PR}(X, Y) = \frac{d_{\mathrm{para}}(X, Y)}{d(X, Y)}. \quad (1)$$

In this formula, $d_{\mathrm{para}}(X, Y)$ denotes the number of documents satisfying the above conditions, and $d(X, Y)$ presents the number of documents having the parenthetical expression $X(Y)$. The function PR(X, Y) ranges from 0 (no abbreviation instance) to 1 (all parenthetical expressions introduce the abbreviation).

Similarity of local contexts We regard words that have dependency relations from/to the target expression as the local contexts of the expression, applying all sentences to a dependency parser (Kudo and Matsumoto, 2002). Collecting the local context of the target expressions, we compute the skew divergence (Lee, 2001), which is a weighted version of Kullback-Leibler (KL) divergence, to measure the resemblance of probability distributions P and Q:

$$\mathrm{SKEW}_{\alpha}(P||Q) = \mathrm{KL}(P||\alpha Q + (1-\alpha)P), \quad (2)$$

$$\mathrm{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \quad (3)$$

In these formulas, P is the probability distribution function of the words in the local context for the expression X, Q is for Y, and α is a skew parameter set to 0.99. The function $\mathrm{SKEW}_{\alpha}(P||Q)$ becomes close to zero if the probability distributions of local contexts for the expressions X and Y are similar.

Other features In addition, we designed twelve features for abbreviation recognition: five features, $\mathrm{freq}(X)$, $\mathrm{freq}(Y)$, $\mathrm{freq}(X, Y)$, $\chi^2(X, Y)$, and $\mathrm{LLR}(X, Y)$ to measure the co-occurrence strength of the expressions X and Y (Hisamitsu and Niwa, 2001), $\mathrm{match}(X, Y)$ feature to test whether or not all letters in an abbreviation appear in its full form, three features *letter type*, *first letter*, and *last letter* corresponding to rules about letter types in abbreviation definitions, and three features *POS tags*, *POS categories*, and *NE tags* to utilize information from a morphological analyzer and named-entity tagger (Kudo and Matsumoto, 2002).

4 Evaluation

4.1 Results

We built a system for Japanese abbreviation recognition by using the LIBSVM implementation[2] with a

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

Group	Recall
Acronym	94.4%
Acronym with translation	97.4%
Alias	81.4%
Total	87.6%

Table 4: Recall for each role of parentheses

linear kernel, which obtained the best result through experiments. The performance was measured under a ten-fold cross-validation on the corpus built in the survey, which contains 1,430 abbreviation instances and 6,457 non-abbreviation instances.

The proposed method achieved 95.7% accuracy, 90.0% precision, and 87.6% recall for recognizing Japanese abbreviations. We cannot compare this performance directly with the previous work because of the differences in the task design and corpus. For reference, Yamamoto (2002) reported 66% precision (he did not provide the recall value) for a similar task: the acquisition of lexical paraphrase from Japanese newspaper articles.

Table 4 reports the recall value for each group of abbreviations. This analysis shows the distribution of abbreviations unrecognized by the proposed method. Japanese acronyms, acronyms with translation, and aliases were recognized at 94.4%, 97.4%, and 81.4% recall respectively. It is interesting to see that the proposed method could extract acronyms with translation and aliases even though we did not use any bilingual dictionaries.

4.2 Analyses for individual features

The numerical and boolean features are monotone increasing functions (decreasing for the SKEW feature) as two expressions X and Y are more likely to present an abbreviation definition. For example, the more authors introduce a paraphrase $X \rightarrow Y$, the higher the value that $\mathrm{PR}(X, Y)$ feature yields. Thus, we emulate a simple classifier for each feature that labels a candidate of abbreviation definition as a positive instance only if the feature value is higher than a given threshold θ, e.g., $\mathrm{PR}(X, Y) > 0.9$. Figure 2 shows the precision–recall curve for each feature with variable thresholds.

The paraphrase ratio (PR) feature outperformed other features with a wide margin: the precision and recall values for the best F1 score were 66.2% and 48.1% respectively. Although the performance of this feature alone was far inferior to the proposed method, to some extent Formula 1 estimated actual occurrences of abbreviation definitions.

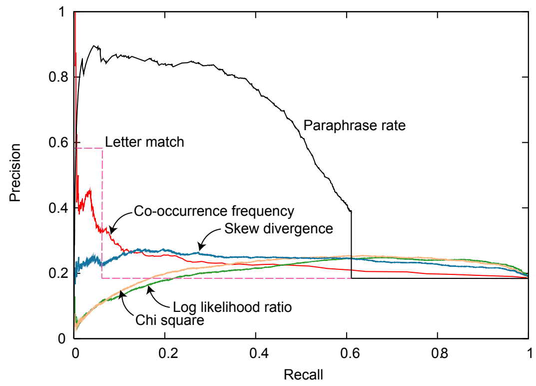


Figure 2: Precision–recall curve of each feature

Feature	Accuracy	Reduction
All	95.7%	—
- $\mathrm{PR}(X, Y)$	95.2%	0.5%
- $\mathrm{SKEW}(X, Y)$	95.4%	0.3%
- $\mathrm{freq}(X, Y)$	95.6%	0.1%
- $\chi^2(X, Y)$	95.6%	0.1%
- $\mathrm{LLR}(X, Y)$	95.3%	0.4%
- $\mathrm{match}(X, Y)$	95.5%	0.2%
- Letter type	94.5%	1.2%
- POS tags	95.6%	0.1%
- NE tags	95.7%	0.0%

Table 5: Contribution of the features

The performance of the match (letter inclusion) feature was as low as 58.2% precision and 6.9% recall[3]. It is not surprising that the match feature had quite a low recall, because of the ratio of 'authentic' acronyms (about 6%) in the corpus. However, the match feature did not gain a good precision either. Examining false cases, we found that this feature could not discriminate cases where an outer element contains its inner element accidentally; e.g., *Tokyo Daigaku (Tokyo)*, which describes a university name followed by its location (prefecture) name.

Finally, we examined the contribution of each feature by eliminating a feature one by one. If a feature was important for recognizing abbreviations, the absence of the feature would drop the accuracy. Each row in Table 5 presents an eliminated feature, the accuracy without the feature, and the reduction of

[3]This feature drew the precision–recall locus in a stepping shape because of its discrete values (0 or 1).

the accuracy. Unfortunately, the accuracy reductions were so few that we could not discuss contributions of features with statistical significance. The letter type feature had the largest influence (1.2%) on the recognition task, followed by the paraphrase ratio (0.5%) and log likelihood ratio (0.4%).

5 Conclusion

In this paper we addressed the difficulties in recognizing Japanese abbreviations by examining actual usages of parenthetical expressions in newspaper articles. We also presented the discriminative approach to Japanese abbreviation recognition, which achieved 95.7% accuracy, 90.0% precision, and 87.6% recall on the evaluation corpus. A future direction of this study would be to apply the proposed method to other non-alphabetical languages, which may have similar difficulties in modeling the generative process of abbreviations. We also plan to extend this approach to the Web documents.

Acknowledgments

This work was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (MEXT, Japan), and Solution-Oriented Research for Science and Technology (JST, Japan). We used *Mainichi Shinbun* and *Yomiuri Shinbun* newspaper articles for the evaluation corpus.

References

Eytan Adar. 2004. SaRAD: A simple and robust abbreviation dictionary. *Bioinformatics*, 20(4):527–533.

Jeffrey T. Chang and Hinrich Schütze. 2006. Abbreviations in biomedical text. In S. Ananiadou and J. McNaught, editors, *Text Mining for Biology and Biomedicine*, pages 99–119. Artech House, Inc.

Jing-Shin Chang and Wei-Lun Teng. 2006. Mining atomic chinese abbreviation pairs: A probabilistic model for single character word recovery. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 17–24, Sydney, Australia, July. Association for Computational Linguistics.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Toru Hisamitsu and Yoshiki Niwa. 2001. Extracting useful terms from parenthetical expression by combining simple rules and statistical measures: A comparative evaluation of bigram statistics. In Didier Bourigault, Christian Jacquemin, and Marie-C L'Homme, editors, *Recent Advances in Computational Terminology*, pages 209–224. John Benjamins.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the CoNLL 2002 (COLING 2002 Post-Conference Workshops)*, pages 63–69.

Lillian Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to wordnet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244.

David Nadeau and Peter D. Turney. 2005. A supervised learning approach to acronym identification. In *8th Canadian Conference on Artificial Intelligence (AI'2005) (LNAI 3501)*, pages 319–329.

Naoaki Okazaki and Sophia Ananiadou. 2006. A term recognition approach to acronym recognition. In *Proceedings of the COLING-ACL 2006 Main Conference Poster Sessions*, pages 643–650, Sydney, Australia.

Serguei Pakhomov. 2002. Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts. In *Proceedings of 40th annual meeting of ACL*, pages 160–167.

Youngja Park and Roy J. Byrd. 2001. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the EMNLP 2001*, pages 126–133.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2007. Improving coreference resolution using bridging reference resolution and automatically acquired synonyms. In *Anaphora: Analysis, Alogorithms and Applications, 6th Discourse Anaphora and Anaphor Resolution Colloquium, DAARC2007*, pages 125–136.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing (PSB 2003)*, number 8, pages 451–462.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Kazuhide Yamamoto. 2002. Acquisition of lexical paraphrases from texts. In *2nd International Workshop on Computational Terminology (Computerm 2002, in conjunction with COLING 2002)*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Linguistic Interpretation of Emotions for Affect Sensing from Text

Mostafa Al Masum Shaikh
Dept. of Information and Communication Engineering
University of Tokyo
7-3-1 Hongo, Bunkyo-Ku
113-8656 Tokyo, Japan
almasum@gmail.com

Helmut Prendinger
Digital Contents and Media Sciences Research Division
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda Ku
101-8430 Tokyo, Japan
helmut@nii.ac.jp

Mitsuru Ishizuka
Dept. of Information and Communication Engineering
University of Tokyo
7-3-1 Hongo, Bunkyo-Ku
113-8656 Tokyo, Japan
ishizuka@ieee.org

Abstract

Several approaches have already been employed to "sense" affective information from text, but none of those ever considered the cognitive and appraisal structure of individual emotions. Hence this paper aims at interpreting the cognitive theory of emotions known as the OCC emotion model, from a linguistic standpoint. The paper provides rules for the OCC emotion types for the task of sensing affective information from text. Since the OCC emotions are associated with several cognitive variables, we explain how the values could be assigned to those by analyzing and processing natural language components. Empirical results indicate that our system outperforms another state-of-the-art system.

1 Introduction and Motivation

While various conceptual models, computational methods, techniques, and tools are reported in (Shanahan et. al., 2006), we argue that the current work for sensing the affect communicated by text is incomplete and often gives inaccurate results. It is true that the assessment of affective content is inevitably subjective and subject to considerable disagreement. Yet the interest in sentiment or affect based text categorization is increasing with the large amount of text becoming available on the Internet. A brief discussion on available approaches is given in (Shaikh et. al., 2007a; Liu et. al., 2003). For example, keyword spotting, lexical affinity, statistical and hand crafted approaches target affective lexicons which are not sufficient to recognize affective information from text, because according to a linguistic survey (Pennebaker et. al., 2003), only 4% of words used in written texts carry affective content.

In this paper we consider the contextual-valenced based approached (i.e., SenseNet) as discussed by Shaikh et. al., (2007a, 2007b) and consider their SenseNet as the basis of our knowledgebase. For simplicity, we use the words 'sentiment' and 'opinion' synonymously and consider sentiment sensing as the prior task of "affect" or "emotion" sensing. The SenseNet can sense either positive or negative "sentiment", but it cannot classify different emotions. Therefore, this paper explains how the SenseNet can be employed to sense emotions from text. So the primary focus of this paper is to provide a set of rules for emotions characterized by the OCC (Ortony et. al., 1988) emotion model and discuss how the rules are implemented.

2 Affect Sensing from Text

2.1 Extending Valence Assignment Approach for Emotions Classification

For the task of affect sensing from text we should incorporate both commonsense knowledge and cognitive structure of emotions along with the semantic interpretation of the words used in a sentence. We have chosen the OCC model of emotions for this task. The rule-based definition of the OCC emotion types characterized by a rich set of linguistic tokens makes it appropriate to cope with the valence assignment approach for affect sensing from text.

2.2 Characterization of OCC Emotions

The OCC emotion types can be characterized by appropriate rules interplaying with several variables. There are two kinds of variables, namely,

emotion inducing variables (event, agent and object based variables) and emotion intensity variables. The event-based variables are calculated with respect to the event which is usually a verb-object pair found in the sentence. For example, the sentence, *John bought Mary an ice-cream*, gives an event as "buy, ice-cream". The variables are enlisted in Table 1. In general we call them "emotion variables".

Type	Variable Name
agent based	agent_fondness ***(af)***
	cognitive_strength ***(cs)***
object based	object_fondness ***(of)***
	object_appealing ***(oa)***
event based	self_reaction ***(sr)***
	self_presumption ***(sp)***
	other_presumption ***(op)***
	prospect ***(pros)***
	status ***(stat)***
	unexpectedness ***(unexp)***
	self_appraisal ***(sa)***
	valenced_reaction ***(vr)***
intensity	event_deservingness ***(ed)***
	effort_of_action ***(eoa)***
	expected_deviation ***(edev)***
	event_familiarity ***(ef)***

Table 1. OCC emotion variables

The OCC emotion model specifies 22 emotion types and 2 cognitive states. For example, OCC model literally defines "Happy-for" as "Pleased about a Desirable event for a Liked agent", and "Fear" as "Displeased about Negative Prospect of an Undesirable Unconfirmed event". Our goal is to represent these literal definitions by rules interplaying with the emotion variables so that the system can evaluate and get either a 'true' or 'false' value. For example, we have an input text *txt*, that has an agent *a*, associated with an event *e*, and we have a program entity *x* that detects emotion from *txt*. We can now represent the rule for "Happy-for" emotion as, *x* senses "Happy-for" if the following condition holds.

[Linguisitc_Token_found_for_HappyFor(txt) and *No_Negation_Found (txt)*] or [***vr*** =True and ***sr*** *(e, txt)* = "Pleased" and ***op***(*e*, *txt*) = "Desirable" and ***af*** (*x*, *txt*) = "Liked" and ***cs*** (a,*x*) = "Other"]

3 Implementation of the Rules

In this section, we first briefly discuss about the SenseNet and its different linguistic resources. Then we explain the 'emotion variables', their enumerated values and how the values are assigned to the respective variables.

3.1 SenseNet

Semantic Parser. The SenseNet has implemented a semantic parser using Machinese Syntax (Connexor Oy, 2005) that produces XML-formatted syntactic output for an input text. For example, the sentence, *"My mother presented me a nice wrist watch on my birthday and made delicious pancakes."*, the output of the semantic parser is shown in Table 2.

Triplet Output of Semantic Parser	
Triplet 1	[[*'Subject Name:'*, 'mother', *'Subject Type:'*, 'Person', *'Subject Attrib:'*, ['PRON PERS GEN SG1:i']], [*'Action Name:'*, 'present', *'Action Status:'*, 'Past ', *'Action Attrib:'*, ['time: my birthday', 'Dependency: and']], [*'Object Name:'*, 'watch', *'Object Type:'*, 'N NOM SG', *'Object Attrib:'*, ['Determiner: a', 'A ABS: nice', 'N NOM SG: wrist', 'Goal: i']]]
Triplet 2	[[*'Subject Name:'*, 'mother', *'Subject Type:'*, 'Person', *'Subject Attrib:'*, []], [*'Action Name:'*, 'make', *'Action Status:'*, 'Past ', *'Action Attrib:'*, []], [*'Object Name:'*, 'pancake', *'Object Type:'*, 'N NOM PL', *'Object Attrib:'*, ['A ABS: delicious']]]

Table2. Semantic Verb-Frames outputted by Semantic Parser

Semantic parser outputs each semantic verb-frame of a sentence as a triplet of "subject, verb, and object". Hence, one obtains multiple triplets if the parser encounters multiple verbs in a sentence. In our case, we consider each triplet to indicate an event encoding the information about "who is doing what and how". Therefore, the output given in Table 2 has two events, which are dependent to each other as indicated by 'dependency' keyword in the action attribute of Triplet 1.

Valenced Output. SenseNet is the implementation of contextual valence based approach that deals with semantic relationships of the words in a sentence and assign contextual-valence using a set of rules and prior-valence of the words. It outputs a numerical value ranging from -15 to +15 flagged as the 'sentence-valence' for each input sentence.

For examples, SenseNet outputs -11.158 and +10.973 for the inputs, *"The attack killed three innocent civilians."* and *"It is difficult to take bad photo with this camera."*, respectively. These values indicate a numerical measure of negative or positive sentiments carried by the sentences.

Scored-list of Action, Adjective, and Adverb. SenseNet has initially assigned prior-valence to 928 verbs, 948 adjectives and 144 adverbs by manual investigations of eight judges where the inter-agreement among the judges are reported as reliable (i.e., the Kappa value is 0.914). The judges have manually counted the number of positive and negative senses of each word of a selected list according to the contextual explanations of each sense found in WordNet 2.1. A database of words with prior-valence assigned using Equations (1) to (3) is developed and scores are stored in the scale of -5 to 5.

Prior-Valence = Average (((Positive-Sense Count – Negative-Sense Count)/Total Sense Count) * 5.0) **(1)**

Prospect Polarity = if (Positive-Sense Count > Negative-Sense Count) then 1 else -1 **(2)**

Prospective Valence = Average(max(Positive-Sense Count, Negative-Sense Count)/Total Sense Count) * 5.0*Prospect Polarity)

Praiseworthy Valence = Average (Prior-Valence + Prospective Valence) **(3)**

Scored-list of Nouns. SenseNet does an automatic approach to assign prior-valence to nouns by employing ConceptNet (Liu and Singh, 2004). A value between -5 to 5 is assigned as the valence for an unrated noun or concept as follows. To assign a prior-valence to a concept, the system collects all semantically connected entities that ConceptNet returns for the input concept. For example, to get the prior-valence for the noun 'rocket', the system failed to find it in the existing knowledgebase, but from the action list of the concept the system returned the value 4.112 by averaging the scores of the verbs 'carry (4.438)', 'contain (4.167)', 'fly (3.036)', 'launch (5.00)' and 'go (3.917)'.

3.2 Assigning Values to the Emotion Variables

According to the OCC model, the values for the variables self_presumption (*sp*) and self_reaction (*sr*) are "Desirable" or "Undesirable", and "Pleased" or "Displeased" respectively. For example, for the events *"buy ice-cream"*, *"present wrist watch"*, *"kill innocent civilians"* referred in the example sentences SenseNet returns contextual valence as +7.832, +8.817 and -8.458, respectively. According to SenseNet scoring system the valence range for an event (i.e., verb, object pair) is ±10. Thereby we decide that for an event if the valence is positive (i.e., "buy ice-cream"), *sp* and *sr* are set as "Desirable" and "Pleased", and in the case of negative valence (i.e., "Kill innocent civilian") *sp* and *sr* are set to "Undesirable" and "Displeased", respectively.

The values for other_presumption (*op*) could be set "Desirable" or "Undesirable". For the sentence *"A terrorist escaped from the Jail"*, the value for *op* (for the event *"escape from jail")* is presumably "Desirable" for the agent "terrorist" but it gets "Undesirable" and "Displeased" for *sp* and *sr* because of negative valence (i.e., -6.715) of the event. From SenseNet we get the valence for terrorist as -3.620. Thus in this case we set *op* as "Desirable" because of having a negative valenced event associated with a negative valenced agent. Similarly we have the following simple rules to assign the values to *op*.

- If a positive valenced event is associated with a positive valenced agent, *op* is set "Desirable". e.g., *the Teacher was awarded the best-teacher award.* [(teacher, +4.167) , (award best-teacher award, +8.741)]
- If a negative valenced event is associated with a positive valenced agent, *op* is set "Undesirable". e.g., *the employee was sacked from the job.* [(employee, +3.445), (sack from job, -6.981)]
- If a positive valenced event is associated with a negative valenced agent, *op* is set "Undesirable". e.g., *the criminal was punished for the crime*. [(criminal,-3.095), (punish for crime, +5.591)]

In this context and in accordance to the OCC model, the value for cognitive_strength (*cs*) indicates how closely the computer program considers selfness. This value is set as "Self" if the agent described in the text is a first person (i.e., I or We); otherwise it is set as "Other". For the sentence, *"I wish I could win the lottery."*, *cs* is set "Self", but for the sentence, *"Susan won the million dollar lottery."*, *cs* is set "Other".

According to the OCC model, prospect of an event involves a conscious expectation that it will

occur in the future, and the value for the variable prospect (*pros*) can be either "Positive" or "Negative". In the aforementioned equation (2), SenseNet considers either the positive or negative sense-count (whichever is the maximum for a verb) to calculate "prospective valence" with the notion of semantic orientation towards optimistic-pessimistic scale. In order to assign *pros* value to an event we also consider the 'prospective valence' of the verb instead of 'prior-valence' of that verb. Thus "positive" or "negative" is assigned according to a certain threshold (i.e., ±3.5) for "positive" or "negative" valence obtained for that event. For example, the events *"admit into university"*, *"kill innocent people"*, *"do it"*, SenseNet returns +9.375, -8.728, +2.921, respectively and according to this valence, *pros* of the events is set to "positive", "negative" and "null", respectively.

The variable status (*stat*) has the values like: "Unconfirmed", "Confirmed" and "Disconfirmed". We decide if the tense of the verb is present or future, the value is set to "Unconfirmed" (e.g., *I am trying to solve it.*); and if it is past or modal without a negation, *stat* is set "Confirmed" (e.g., *I succeeded.*), but with a negation, *stat* is set "Disconfirmed" (e.g., *I did not succeed.*).

If the valence of the agent/object is positive, "Liked" is set to the variables agent_fondness (*af*) and object_fondness (*of*) variables, otherwise "Not-liked" is set. For example, for the sentences, *"The hero appeared to save the girl."*, and *"A terrorist escaped from the Jail"*, *af* for "hero" and "terrorist" is set to "Liked" and "Not-Liked" because of positive and negative valence. Similarly, *of* is set "Liked" and "Not-Liked" for "girl" and "Jail" respectively.

The value for self_appraisal (*sa*) can be either "Praiseworthy" or "Blameworthy". In the aforementioned equation (3) SenseNet takes the average of "Prior Valence" and "Prospective Valence" of a verb with the notion of average semantic orientation of the verb from both good-bad and optimistic-pessimistic perspective. Like assigning *pros* value to an event we consider the "praiseworthy valence" of the verb to assign value to *sa*. Thereby for the same events discussed above to explain *pros* assignment, the value for *sa* is set "Praiseworthy", "Blameworthy" and "null", respectively.

The value of object_appealing (*oa*) indicates whether an object is "Attractive" or "Unattractive". In order to assign a value to *oa*, we deal with two scores (i.e., object valence, and familiarity valence) having the following heuristic. "Attractive" is set if the object has a positive valence with a familiarity valence less than a certain threshold. Reversely "Unattractive" is set if the object has a negative valence with a familiarity valence above a certain threshold. The familiarity valence is obtained from the ConceptNet by calculating the percentage of nodes (out of 300,000 concept-nodes) linking to and from the given object/concept. For example, the familiarity valence for "restaurant", "thief" and "diamond ring" is 0.242%, 0.120% and 0.013%, respectively. Heuristically we kept the threshold 0.10% to signal familiarity and unfamiliarity of an object. Thus "diamond ring" and "thief" gets "Attractive" and "Unattractive" set for *oa*, but "restaurant" gets 'null' accordingly.

The value for valenced_reaction (*vr*) is set either "True" or "False" in order to initiate further analysis to sense emotions or decide the sentence(s) as expressing a neutral emotion. We consider *vr* to be "True" if the 'sentence-valence' returned by SenseNet is either above than 3.5 or less than -3.5. For example, *"I go."*, doesn't lead to further processing (i.e., sentence-valence is +3.250) but *"I go to gym everyday."*, leads to classify emotion because of the sentence-valence +7.351 obtained from SenseNet. The value to the variable unexpectedness (*unexp*) is set "true" if there is a linguistic token to represent suddenness (e.g., abruptly, suddenly, swiftly etc.) in the input sentence, otherwise "false" is set. We have a list of such tokens to indicate suddenness.

OCC model has several variables to signify emotional intensity. For example, the value for the intensity variable event_deservingness (*ed*) is set "High" for an event having a higher positive valence (i.e., above +7.0) or "Low" for higher negative one (i.e., less than -7.0). If an action is qualified with an adverb (e.g., *He worked very hard*) or target object qualified with an adjective (e.g., *I am looking for a quiet place*) without a negation, the value for effort_of_action (*eoa*) is set "Obvious", otherwise "Not-Obvious". Another variable called expected_deviation (*edev*) indicates the difference between the event and its actor. For example, in the sentence *"The police caught the criminal finally."*, the actor "police" and the event "catch criminal" don't deviate because the action is presumably expected by the actor. We set the value for *edev* to "Low" if ConceptNet can find any se-

mantic relationship between the actor and event; otherwise "High" is set. For example, for sentence *"the student invented the theory."*, *edev* is set "High" because ConceptNet doesn't return any relationship between "student" and "invent". The values "Common" or "Uncommon" are set for event_familiarity (*ef*) according to the familiarity valence obtained from ConceptNet for the input event as discussed before.

4.3 The rules for the OCC Emotion Types

In section 2.2 we briefly illustrated how a rule for the OCC defined emotion (e.g., happy-for) is characterized. Now using the same notion we enlist the rules for the OCC model defined emotion types. Although in *txt* there might be multiple *e* described and we also deal with such cases to get the resultant emotion types from *txt*, but we don't discuss that in the scope of this paper and describe the simple cases. Thus, the rules for emotion types are given considering an event *e*, for example, the program *x* senses 'Joy' for *e* if following condition is true:

[Linguisitc_Token_found_for_Joy(txt) and *No_Negation_Found (txt)*] or [***vr***= true and ***sr***= "Pleased" and ***sp***= "Desirable" and ***cs***= "Self"] (i.e., literally joy means being 'pleased about a desirable event'.) Since we have the token words for each emotion types, we omit the first condition in the subsequent rules for space limitations. The rules for the emotion are listed as following and due to space limitations we are not providing the rules for all the emotions.

- if (***vr***= true and ***sr***= "Pleased" and ***pros***= "Positive" and ***sp***= "Desirable" and ***status***= "Unconfirmed"), "hope" is true.
- if (***vr=*** true and ***sr*** = "Displeased" and ***pros***= "Negative" and ***sp***= "Undesirable" and ***status***="Unconfirmed"), "fear" is true.
- if (***vr=*** true and ***sr*** = "Pleased" and ***pros***= "Negative" and ***sp***= "Undesirable" and ***status***= "Disconfirmed"), "relief" is true.
- if (***vr=*** true and ***sr*** = "Displeased" and ***pros***= "Positive" and ***sp***= "Desirable" and ***status***= "Disconfirmed"), "disappointment" is true.
- if (***vr=*** true and ***sr***= "Displeased" and ***sa***= "Blameworthy" and ***sp***= "Undesirable" and ***cs***="Self"), "shame" is true.
- if (***vr=*** true and ***sp***= "Desirable" and ***sr***= "Pleased" and ***of***= "Liked" and ***oa***= "Attractive"), "love" is true.
- if (***vr=*** true and ***sp***= "Undesirable" and ***sr***= "Displeased" and ***of***= "Not-Liked" and ***oa***= "Unattractive"), "hate" is true.

The OCC model has four complex emotions namely, "gratification", "remorse", "gratitude" and "anger". For example:

- If both "joy" and "pride" are true, "gratification" is also true.
- If both "distress" and "reproach" are true, "anger" is also true.

The cognitive states "Shock" (i.e.; unpleasant surprise) and "Surprise" (i.e., pleasant surprise) are ruled as; If both "distress" and ***unexp*** are true, "shock" is true. (e.g., The bad news came unexpectedly.). Similarly, if both "joy" and ***unexp*** are true, "surprise" is true. (e.g., I suddenly met my school friend in Tokyo.)

Like Liu et al. (2003), we also believe that a statement may contain more than one type of emotions. In our case, the 22 emotion types and two cognitive states are grouped into seven groups, namely, well-being emotion, fortune of other emotion, prospect based emotion, cognitive state, attribution emotion, attraction emotion, and compound emotion. Hence an input sentence may contain one of the emotion types from each group. For example, the sentence *"I suddenly got to know that my paper won the best paper award."*, outputs the following emotions: {Joy, Satisfaction, Surprise, Pride, Gratification}.The sentence *"She failed to pass the entrance examination."*, outputs {Distress, Sorry-for, Disappointment, Reproach, Anger} emotion types. In order to reduce the number of emotions, we consider the intensity variables. For the first set of emotions, we can reduce it to {Satisfaction, Surprise, Pride} because "Joy" doesn't have any intensity variables and the intensity variables *ed* and *edev* are set to "High" in this case.

4 Test and Evaluation

The similar system like ours is Liu's system (Liu et. al., 2003). It is a rule based system, and it seems to be the best performing system for sentence-level affect sensing that senses happy, fearful, sad, angry, disgust, and surprise emotions. On the practical side, it is freely available on the Internet. Ex-

ample input and output are enlisted to given an idea about the outputs of the two systems.

Input: I avoided the accident luckily.

Liu's output: fearful(26%),happy (18%), angry (12%),sad(8%),surprised(7%),disgusted (0%)

Ours output: valence: +11.453; [joy, pride, relief, surprise, gratification]

Input: Susan bought a lottery ticket and she was lucky to win the million dollar lottery.

Liu's output: sad(21%), happy(18%), fearful (13%),angry(11%),disgusted(0%),surprised (0%)

Ours: valence: +12.533; [happy-for, satisfaction, admiration, love]

We evaluated our system to assess the accuracy of sentence-level affect sensing when compared to human-ranked scores (as "gold standard") for 200 sentences assessed by two systems. The sentences were collected from Internet based sources for reviews of products, movies, and news. In order to conduct system's performance and acceptance test we have two systems X (i.e., Liu's System) and Y (i.e., our system). The judges were not told about the characteristics of any of the systems. Each judge receives the output from both X and Y for each input sentence and can accept either both outputs or anyone of the two or reject both. Thus %X means the percentage of the number of acceptances received by X in terms of accuracy of output. Similarly %Y, %XY, and %!XY indicate the percentage of acceptances received by the system Y, both the systems and neither of the two systems respectively. For example, for the input sentence *"She is extremely generous, but not very tolerant with people who don't agree with her."*, among the 5 judges 3 accepted the output of Y, 2 accepted the output of X. Since the majority of the judges accepted Y, vote for this sentence was counter for Y. Thus the vote for each sentence is counted. Outcome of our experiment is reported below while the valence range to classify a neutral sentence is considered ±3.5 for the SenseNet upon which system Y is built.

System Y received 16.069% more acceptances than that of X, which indicates that the output of Y is more acceptable and accurate than that of X. Though the test was conducted with a small group of judges with relatively small input size, but the experiment result (i.e., 82% accuracy with an average precision 76.49%, recall 81.04% and F-score 78% for classifying positive, negative and neutral classes using the same dataset) for sentiment sensing reported by SenseNet, provides an optimistic believe that the result would not vary even the survey is conducted with larger group of judges. Table 3 summarizes the experimental result for 200 sentences.

Data-Set of 200 Sentences			
%X	%Y	%XY	%!XY
20.344	36.413	24.283	18.96

Table 3. Experimental Result

5 Conclusion

In order to perform more testing and usability study, we plan to implement a web-based user interface where any user can input a chunk of text and get outputs from the both systems mentioned above. Thereby we can get user's acceptance test in terms of accuracy of output. Next we plan to perform the task of affect sensing using online resources (e.g., blogs, reviews, etc.).

Reference

Connexor Oy. 2005. *Machinese Syntax*, web-site: http://www.connexor.com/connexor/

Hugo Liu and Push Singh. 2004. ConceptNet: A Practical Commonsense Reasoning Toolkit, *BT Technology Journal,* 22(4):211-226, Kluwer Academic Publishers.

Hugo Liu, Henry Lieberman, and Ted Selker. 2003. A Model of Textual Affect Sensing using Real-World Knowledge, In *Proc. IUI 03*, pp. 125-132, Miami, USA, ACM.

Opinmind, Discovering Bloggers, (2006), http://www.opinmind.com/

Andrew Ortony, Gerald L. Clore and Allan Collins. 1988. The Cognitive Structure of Emotions, Cambridge University Press.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. Linguistic inquiry and word count: *LIWC* (2nd ed.) [Computer software]. Mahwah, NJ: Erlbaum.

Mostafa A. M. Shaikh, Helmut Prendinger and Mitsuru Ishizuka. 2007a. SenseNet: A Linguistic Tool to Visualize Numerical-Valance Based Sentiment of Textual Data, In *Proc.* ICON-2007, pages 147-152.

Mostafa A. M. Shaikh, Helmut Prendinger and Mitsuru Ishizuka. 2007b. Assessing Sentiment of Text by Semantic Dependency and Contextual Valence Analysis. In *Proc. ACII 07,* pp. 191-202.

James G. Shanahan, Yan Qu and Janyce Wiebe (Eds.). 2006. Computing Attitude and Affect in Text: Theory and Applications, Springer.

How to Take Advantage of the Limitations with Markov Clustering?

The Foundations of Branching Markov Clustering (BMCL)

Hiroyuki Akama
Tokyo Institute of Technology
W9-10, 2-12-1,
O-okayama, Meguroku,
152-8552 Tokyo, Japan
akama.h.aa@m.titech.ac.jp

Maki Miyake
University of Osaka
Machikane-machi, Toyo-naka-shi,
560-0043 Osaka, Japan
mmiyake@lang.osaka-u.ac.jp

Jaeyoung Jung
Tokyo Institute of Technology
W9-10, 2-12-1,
O-okayama, Meguroku,
152-8552 Tokyo, Japan
jung.j.aa@m.titech.ac.jp

Abstract

In this paper, we propose a novel approach to optimally employing the MCL (Markov Cluster Algorithm) by "neutralizing" the trivial disadvantages acknowledged by its original proposer. Our BMCL (Branching Markov Clustering) algorithm makes it possible to subdivide a large core cluster into appropriately resized sub-graphs. Utilizing three corpora, we examine the effects of the BMCL which varies according to the curvature (clustering coefficient) of a hub in a network.

1 MCL limitations?

1.1 MCL and modularity Q

The Markov Cluster Algorithm (MCL) (Van Dongen, 2000) is well-recognized as an effective method of graph clustering. It involves changing the values of a transition matrix toward either 0 or 1 at each step in a random walk until the stochastic condition is satisfied. When the hadamard power for each transition probability value is divided by the sum of each column, the rescaling process yields a transition matrix for the next stage. After repeatedly alternating for about 20 times between two steps—random walk (*expansion*) and probability modification (*inflation*)—the process will finally reach a convergence stage in which the whole graph is subdivided into a set of 'hard' clusters that have no overlap. Although this method has been generally applied in various domains with notable successes (such as Tribe-MCL clustering of proteins (Enright et al., 2002); Synonymy Network, created by the addition of noise data (Gfeller, 2005); and Lexical Acquisition (Dorow et al., 2005)), Van Dongen et al. (2001) frankly acknowledge that there are limitations or weaknesses. For instance, the readme file, which is included with the free MCL software available via the Internet from Van Dongen's group, remarks that "MCL is probably not suited for clustering tree graphs".

It should also be noted, however, that the group has provided no mathematical evidence for their claim of the MCL's unsuitability for hierarchical applications. What prompts this subtle caveat in the first place? Is this a limitation on the type of graph clustering that can employ random walks for spectral analysis? Or, is it difficult for this technique to (re-)form or adjust graph clusters that have already been clustered into a kind of multi-layered organization? Such questions are very important when comparing the MCL with other graph clustering methods that employ (greedy) algorithms developed step by step in a tree form.

A tree graph is essentially a kind of dendrogram, which means clustering results can be generated solely by making a cross cut at some height between the root and the leaves. In other words, as there is no horizontal connection at the same level, it is not possible to create triangle circulation paths in a single stroke. However, the graph coefficient known as "curvature" (Dorow, 2005) is appropriate for defining such structures. The curvature, or the cluster coefficient, of a vertex is defined as a fraction of existing links among a node's neighbors out of all possible links between neighbors. Thus, a tree graph may be regarded as a chain of star graphs where all the vertices have a curvature value of 0.

It is certainly true that when a hub has a low curvature value, the corresponding cluster will be less cohesive and more sparse than usual. The modularity Q value is very low in such cases when we try to measure the accuracy of results from MCL clustering. Modularity Q indicates differences in edge distributions between a graph with meaningful partitions and a random graph for identical vertices conditions. According to Newman and Girvan, $Q=\sum_i (e_{ii}-a_i^2)$, where i is the cluster number of cluster c_i, e_{ii} is the proportion of internal links in the whole graph and a_i is the expected proportion of c_i's edges calculated as the total number of degrees in c_i divided by the total of all degrees (2*the number of all edges) in the whole graph. This value has been widely used as an index to evaluate the accuracy of clustering results.

1.2 Karate club simulation

However, it would be an exaggeration to regard Modularity Q is an almighty tool for accurately determining the attribution value of each vertex in a graph cluster. That is only true for modularity-based greedy algorithms that select vertices pairings be merged into a cluster at each step of the tree-form integration process based on modularity optimization criterion. However, such methods suffer from the problem that once a merger is executed based on a discrimination error, there is no chance of subsequently splitting pairings that belong to different subgroups.

This fatal error can be illustrated as follows. Zachary's famous "Karate Club" is often used as supervised data for graph clustering, because the complex relationships among the club members are presented as a graph composed of edges representing acquaintances and vertices coded indicating final attachments to factions. If the results of graph clustering were to match with the actual composition of sects within the club, one could claim that the tested method was capable of simulating the social relationships.

However, the real difficulties lie at boundary positions. It is worth pointing out that the degree of ambiguity is the same (0.5) for both vertices 3 and 10 in Figure I, indicating that they occupy neutral positions while in reality they belong to different subgroups. All modularity-based greedy algorithms would inevitably bind the two nodes at an earlier step in the dendrogram construction (at the first merging step in experiments conducted by Newman and Danon and at the second in Pujol's experiment). In contrast, MCL is one of the rare clustering methods that avoids this type of misjudgment (accurate results for the karate club network were also obtained with the Ward method), even though the modularity Q value for MCL is a little lower (0.371) than values for greedy algorithms (for example, 0.3807 for Newman et al.'s fast algorithm and 0.418 for Danon et al.'s modified algorithm).

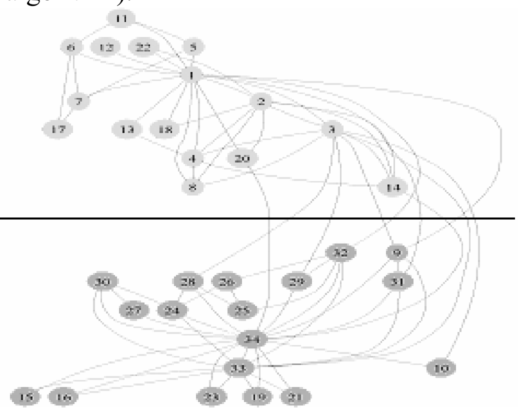
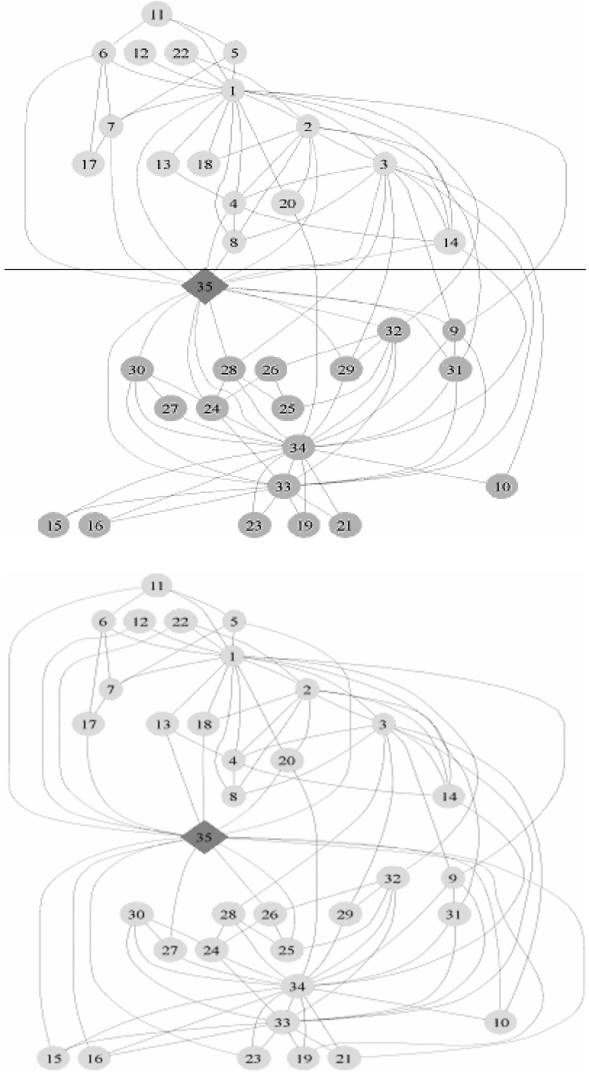


Figure I Karate club

The karate club case suggests the possibility of using both graph clustering and modularity Q from different perspectives. MCL allows us to regard both *clustering* and *discrimination* on the same plan if we do not treat modularity Q as an optimization index but rather as an index of structuring dynamics balancing assembly and division. To the extent that a graph clustering method is evaluated in terms of its effectiveness in a variety of discrimination analyses with learning data extracted from real situations, it should be useful as a simulation tool. For example, it is possible to test with the karate club network the effects of supplementing the network by adding to the original graph another hub with the highest degree value. As the curvature value of this new hub varies according to the selection of vertices which become adjacent to it, we can re-execute MCL for the overall graph to see how curvature is closely related with how it influences clustering results. In general cases, the hub of a whole graph also tends to be the representative node for the large-sized Markov cluster called the "core cluster" (Jung, 2006).

Let us imagine that a highly influential newcomer joins the karate club and tries to contact with half (17) of all the members, functioning as a hub within the network. Even though this is a purely hypothetical situation, it is possible to predict the impact on the network with MCL.



Figures II, III Hub to high or low degree nodes

For example, one could classify the 34 vertices into higher and lower degree subgroups, and set a hub that is adjacent to all vertices for one subgroup but is far from the other subgroup. MCL results would indicate that even when adding a hub with the highest curvature value, it would be ineffectual in preventing a split (Figure II). However, if the newcomer were to be a friend with less sociable members, the club would be saved from being torn apart. A hub connected with the lower degree subgroup, and thus having the lowest curvature value, would become part of the largest core cluster, because the MCL would not subdivide the graph (Figure III). In short, the results of MCL computation hinge on the curvature value of the hub with the highest degree value.

2 The basic concept of BMCL

This connection-sensitive feature of MCL brings us back to the limitations that Van Dogen et al. inform their software users of. Do these limitations really render the MCL unsuitable for tree graphs? Should we not regard a low modularity Q value for a graph as a positive attribute if it is due to the low curvature value for a hub? In a very real sense, these questions are actually asking about the same thing. The point can be clearer if conceived of in relation to a non-directed and cascading type of three-layer graph, as depicted in Figure IV.

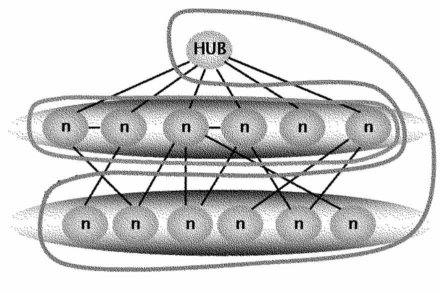


Figure IV Three-layer tree-form network

The root node at the top (the hub) is linked to all the vertices in the intermediate layer but to none at the bottom layer, even though there are moderate levels of connectivity between the layers. Connections within a layer are extremely rare or absent. Clearly, the curvature of the hub would be influenced by the very low connectivity within layer 2.

	0.01	0.02	0.03
0.1	1core cluster & singleton clusters	1core cluster & singleton clusters	1cluster (not divided)
0.15	1cluster or 2 core clusters	1cluster (not divided)	1cluster (not divided)
0.2	2core clusters	2core clusters	1cluster (not divided)

Table I. MCL results for the structured Random Graph

We have executed computations at least 10 times under the same condition in order to generate this type of structured random graph with 500 vertices in the two layers respectively. A random graph was produced by using a binominal distribution. Although between connection rates were varied from 0.1 to 0.2 and within connection rates for the intermediate layer from 0.01 to 0.03, no edges were inserted into the lower layer. MCL results obtained for this architecture are almost constant, as shown in Table I.

In this experiment, all singleton clusters consisted of vertices belonging to layer 2. In cases where the whole graph was split into 2 core clusters, one cluster would correspond to the hub plus layer 3 while the other would correspond to layer 2. There was no exception when the between connection rate was 0.2. This means that, quite curiously, the hub formed a core cluster around itself with vertices that were not all adjacent to it, so that ones that were connected with it in the raw data were all segregated into the other cluster. In this case, the Modularity Q value for each core cluster was zero or extremely low.

Nevertheless, in spite of this inaccuracy, this type of network can easily be by modified by the BMCL method that we discuss later. It can be indirectly subdivided by graph clustering, if inside the same cluster, a latent shortcut is set between one vertex and another. Such a latent connection can be counted in place of a path of length 2 that is traced in the original adjacency as a detour via a vertex of another cluster. If all latent adjacency relationships are enumerated in this way, except for those for the hub, the core cluster will be reclustered by a second application of the MCL to realize a sort of hierarchical clustering (in this case for a quasi-tree graph), which has been regarded as being a limitation with the MCL.

This principle can be called Branching Markov Clustering (BMCL) in the sense that it makes it possible to correct for unbalances in cluster-sizes by dividing large Markov clusters into appropriate branches. In other words, BMCL is a way of rebuilding adjacency relationships "inside" MCL clusters, by making reference to "outside" path information. It then becomes natural to realize that the lower the curvature value of the hub is—reflecting sparse connectivity inside the hub's cluster—the more effective BMCL will be in subdividing the core cluster, which will augment the modularity Q value for the clustering results.

3 Applying BMCL corpora data

3.1 The BMCL algorithm

In this section, we apply our BMCL method to a semantic network that is almost exhaustively extracted from typical documents of a specific structure. It is supposed that if the MCL is applied to word association or co-occurrence data it will yield concept clusters where words are classified according to similar topics or similar meanings as paradigms. However, because the word distribution of a corpus approximately follows Zipf's law and produces a small-world scale-free network (Steyvers et al., 2005), the MCL will result in a biased distribution of cluster sizes, with a few extraordinarily large core clusters that lack any particular features.

In order to overcome such difficulties in building appropriate lexical graphs for corpus data, we propose an original way of appropriately subdividing core clusters by taking into account graph coefficients, especially the curvature of a hub word. As mentioned above, BMCL is most effective for clusters that, containing a high-degree and low-curvature vertex, display a local part of a network with highly sparse connectivity when a hub is eliminated. This feature increases the efficiency of the BMCL by making it possible to introduce moderate connection rates for latent adjacencies.

In contrast to a 'real' adjacency between the vertices i, k represented here by $d(i,k)=1$, the 'latent' adjacency $d_v(i,j)=1$ will subsequently be defined to closely adapt to the connection state for the dataset, which we will utilize in testing the BMCL. The hub M_h of each Markov cluster M is supposed to be the vertex with the largest degree for M. Here, we set a sufficiently large core cluster C, a set of hubs H and the hub of C as C_h. Under such conditions, we can formulize the set of external hubs bypassing the intra-core connections $K_{i,j}$ as;

$\{K_{i,j} \mid K_{i,j} \subset H, k \in K_{i,j}, d(i,k) = d(j,k) = 1\}$,

where $i, j \in \underset{i \neq j, i \neq C_h, j \neq C_h}{C}, C_h \notin H$. We also propose an additional function called $\underset{n}{ArgTopn}$, which identifies the set of n nodes that have the highest connec-

tion values. This is to produce a moderate connection rate which allows us to execute appropriate MCL operations by appropriately setting two pruning thresholds, θ_p and θ_q. These are applied in the row direction by fixing i in the intra-core connection matrix to the number of the shortest paths between i, j --$|K_{i,j}|$-- to make the following pruning rule:

$$if(|K_{i,j}| \geq \theta_p \; \&\& \; j \in \underset{n=\theta_q}{ArgTopn} |K_{i,j}|)$$
$$-> d_v(i,j) = 1$$

This rule extracts from the intra-core connection matrix a latent adjacency matrix to which the MCL is applied once again in order to obtain appropriately resized sub-clusters from a huge core cluster.

3.2 A range of corpus data

In this section, three documents were selected taking into consideration the curvature value of a hub with the highest degree and the density of connections with or without this hub among the vertices of a core cluster at the level of a raw data graph.

I. Associative Concept Dictionary of Japanese Words (Ishizaki et al., 2001), hereafter abbreviated as ACDJ, which consists of 33,018 words and 240,093 word pairing collected in an association task involving 10 participants. Of these, 9,373 critical words were selected to create well-arranged semantic network by removing the rarest 1-degree dangling words and rarer words with a degree of 2 but curvature values of 0.

II. Gakken's Large Dictionary of Japanese (Kindaichi & Ikeda, 1988), hereafter abbreviated as GLDJ, which is an authoritative Japanese dictionary with some features of an encyclopedia in terms of its rich explanatory texts and copious examples. We selected 98,083 words after removing noise words, functional words, and 1,321 isolated words to extract word pairs by combining every headword with every other headword included within an entry text.

III. WordNet. We used only the "data.noun" file where the lexical information for each noun is defined by a set of index numbers corresponding not with words themselves but with their senses. The co-occurrence relationships for 98,794 meanings were extracted from every data block that contains a series of indexes, which also covers other parts-of-speech.

The principle for building a semantic network for each of these documents was to select relevant 'word pairs' or 'index pairs' indicating the lexical relationships of adjacency, association or co-occurrence, respectively. Table II presents graph information for the three data sets and the results of applying both the MCL and the BMCL to them.

	ACDJ	GLDJ	WordNet
Num of Vertices	9373	98083	98794
Degree Mean	19.963	13.8939	63.7155
Hub Word	House	Archaic Words	Individual
Degree of Hub	563	12959	2773
Curvature of Hub	0.0398	8.51106E-05	0.0405
Core Cluster Size	158	8962	2597
Connection Rate of Core Cluster	0.0022	0.000328782	0.030539
Ibid (Without Hub)	0	0.000153119	0.03005
Q for the First MCL	0.0946409	0.176	0.841275
Q for the BMCL	0.606284	0.221	-0.094

Table II Data about the three corpora

Although the first data (ACDJ) is much smaller, it is worthwhile executing because it represents a concrete example of the network type discussed earlier, namely, a three-layer architecture around a hub (quasi-tree graph). The connection rate in the core cluster is very low (0.002 with and 0 without the hub), as is the modularity Q value for the MCL (0.094). However, subdivision of the core cluster in the BMCL results yielded a high modularity Q value (0.606) when latent adjacencies derived from bypassing connections with a threshold of θ_q =3 were used.

The last two data (GLDJ and WordNet) are directly comparable because they are quite similar in size and provide sharp contrast, particularly in terms of curvature values (GLDJ: 8.51106E-05 << WordNet: 0.0405), and modularity Q values for the MCL (GLDJ: 0.176 << WordNet: 0.841). For WordNet, the high connection rate in the core cluster (0.03) makes it difficult for it to be subdivided by any clustering method, even if the hub is eliminated. In terms of the GLDJ, the core cluster was repeatedly divided by the BMCL and the modularity for the subdivision turned out to be 0.2214 with a threshold of θ_p = 1.

However, there is another way to split the core cluster into sub graphs, which does not require the use of the latent adjacency information which is crucial for the BMCL. That other method, which can be called the 'Simply-Repeated MCL (SR-MCL)', involves applying the MCL once again to

the part of the original adjacency matrix that corresponds to the vertices apart from the hub, and which become members of the core cluster as a result of the first MCL. In the case of ACDJ, it is impossible to execute the SR-MCL, because there is no edge that is not connected to the hub within the core cluster, and so all the vertices apart from the hub would be isolated if the hub were removed.

A similar problem is also encountered with the core cluster of the GLDJ, even though the SR-MCL increases the modularity Q value (0.769) much more than the BMCL. Vertices that dangle from the hub—37% of the core members—would be dropped from the second MCL computation if the latent adjacency is not used, which, on the other hand, assures a high recall rate (0.88). Thus, we have adopted an eclectic way to maintain both relatively high *recall* (the proportion of non-isolated nodes) and relatively high *precision* (the modularity Q of the intra-core clustering). This is what we may call a 'Mixed BMCL' which involves combining the latent adjacency matrix exclusively for the vertices dangling to the hub and the raw adjacency part matrix for the remaining ones that are connected among them. As Figure V highlights, the F-measure $\frac{PR}{(1-\alpha)P+\alpha R}$ (R: recall; P: precision) underscores the effectiveness of the Mixed BMCL for the GLDJ.

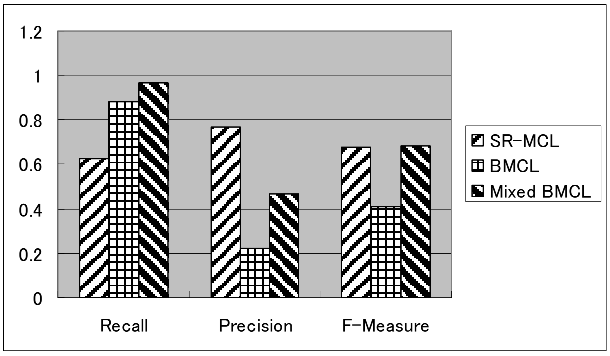


Figure V Comparison of the methods (α =0.4)

4 Conclusion

This paper has examined MCL outputs obtained for some rather problematic conditions, such as the clustering of a tree graph and clustering for a network that contains a hub that has a very low curvature value. In such cases, many of the vertices adjacent to the hub are removed from the cluster that it represents. However, compensating for that, the hub cluster will absorb many other vertices—some of which are not directly connected to the hub itself—to form a large-sized core cluster. That is when our proposed method of Branching MCL (BMCL) is most effective in adjusting cluster sizes by utilizing latent adjacency. Subdivision of the core cluster can facilitate the interpretation of the classified concepts.

When the curvature of the hub is a little higher than in such extreme conditions, the combination of the ordinary MCL and the BMCL (a Mixed BMCL) can work well in increasing the F-Measure score. However, it is not possible to reapply the MCL to a dense core cluster that is organized around a hub with a very high curvature value. A direction for further research will be to automatically select from between the BMCL and the Mixed-BMCL. The SR-MCL or similar modifications may yield the optimal approach to dividing massive Markov clusters into appropriate subsets.

References

Clauset, A, Newman M.E.J., and Moore, C. Finding Community Structure in Very Large Networks, Phys. Rev. E 70, 066111 (2004)

Danon, L., Diaz-Guilera, A., and Arenas, A. Effect of Size Heterogeneity on Community Identification in Complex Networks, J. Stat. Mech. P11010 (2006)

Dorow, B. et al. Using Curvature and Markov Clustering in Graphs for Lexical Acquisition and Word Sence Discrimination, MEANING-2005,2nd Workshop organized by the MEANING Project, February,3rd-4th.(2005)

Kindaichi, H., Ikeda, Y. Gakken's Large Dictionary of Japanese, GAKKEN CO, LTD. (1988)

Newman M. E. J. and Girvan M., Finding and evaluating community structure in networks, Physical Review E 69. 026113, (2004)

Okamoto, J., Ishizaki, S. Associative Concept Dictionary and its Comparison with Electronic Concept Dictionaries, http://afnlp.org/pacling2001/pdf/okamoto.pdf, (2001).

Pujol, J.M., Béjar, J. and Delgado, J. "Clustering Algorithm for Determining Community Structure in Large Networks".Physical Review E 74 (2007):016107

Van Dongen, S. Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht. (2000)

Combining Context Features by Canonical Belief Network for Chinese Part-Of-Speech Tagging

Hongzhi Xu and Chunping Li
School of Software, Tsinghua University
Key Laboratory for Information System Security, Ministry of Education China
xuhz05@mails.tsinghua.edu.cn
cli@tsinghua.edu.cn

Abstract

Part-Of-Speech(POS) tagging is the essential basis of Natural language processing(NLP). In this paper, we present an algorithm that combines a variety of context features, e.g. the POS tags of the words next to the word a that needs to be tagged and the context lexical information of a by Canonical Belief Network to together determine the POS tag of a. Experiments on a Chinese corpus are conducted to compare our algorithm with the standard HMM-based POS tagging and the POS tagging software ICTCLAS3.0. The experimental results show that our algorithm is more effective.

1 Introduction

Part-Of-Speech(POS) tagging is the essential basis of Natural language processing(NLP). It is the process in which each word is assigned to a corresponding POS tag that describes how this word be used in a sentence. Typically, the tags can be syntactic categories, such as noun, verb and so on. For Chinese language, word segmentation must be done before POS tagging, because, different from English sentences, there is no distinct boundary such as white space to separate different words(Sun, 2001). Also, Chinese word segmentation and POS tagging can be done at the same time(Ng, 2004)(Wang, 2006).

There are two main approaches for POS tagging: rule-based and statistical algorithms(Merialdo, 1994). Rule based POS tagging methods extratct rules from training corpus and use these rules to tag new sentences(Brill, 1992)(Brill, 1994). Statistic-based algorithms based on Belief Network(Murphy, 2001) such as Hidden-Markov-Model(HMM)(Cutting, 1992)(Thede, 1999), Lexicalized HMM(Lee, 2000) and Maximal-Entropy model(Ratnaparkhi, 1996) use the statistical information of a manually tagged corpus as background knowledge to tag new sentences. For example, the verb is mostly followed by a noun, an adverb or nothing, so if we are sure that a word a is a verb, we could say the word b following a has a large probability to be a noun. This could be helpful specially when b has a lot of possible POS tags or it is an unknown word.

Formally, this process relates to $Pr(noun|verb)$, $Pr(adverb|verb)$ and $Pr(nothing|verb)$, that can be estimated from the training corpus. HMM-based tagging is mainly based on such statistical information. Lexicalized HMM tagging not only considers the POS tags information to determine whether b is noun, adverb or nothing, but also considers the lexical information a itself. That is, it considers the probabilities $Pr(noun|a, verb)$, $Pr(adverb|a, verb)$ and $Pr(nothing|a, verb)$ for instance. Since combining more context information, Lexicalized HMM tagging gets a better performance(Lee, 2000).

The main problem of Lexicalized HMM is that it suffers from the data sparseness, so parameter smoothing is very important. In this paper, we present a new algorithm that combines several context information, e.g. the POS tags information and lexical information as features by Canonical Belief Network(Turtle, 1991) to together determine the tag

of a new word. The experiments show that our algorithm really performs well. Here, we don't explore Chinese word segmentation methods, and related information can be found in(Sun, 2001).

The rest of the paper is organized as follows. In section 2 and section 3, we describe the standard HMM-based tagging and Lexicalized HMM tagging respectively which are relevant to our algorithm. In section 4, we describe the Belief Network as a preliminary. In section 5, we present our algorithm that is based on Canonical Belief Network. Section 6 is the experiments and their results. In section 7, we have the conclusion and the future work.

2 Standard Hidden Markov Model

The problem of POS tagging can be formally defined as: given an observation(sentence) $w = \{w_1, w_2, ..., w_T\}$ and a POS tag set $TS = \{t_1, t_2, ..., t_M\}$, the task is to find a tag sequence $t = \{t_1, t_2, ..., t_T\}$, where $t_i \in TS$, that is the most possible one to explain the observation. That is to find t to maximize the probability $Pr(t|w)$. It can be rewritten by Bayesian rule as follows.

$$Pr(t|w) = \frac{Pr(w|t) \times Pr(t)}{Pr(w)}$$

As for any sequence t, the probability $Pr(w)$ is constant, we could ignore $Pr(w)$. For $Pr(t)$, it can be decomposed by the chain rule as follows.

$$\begin{aligned} Pr(t) &= Pr(t_1, t_2, ..., t_T) \\ &= Pr(t_1) \times Pr(t_2|t_1) \times Pr(t_3|t_1, t_2) \times \\ &\quad ... \times Pr(t_T|t_1, t_2, ..., t_{T-1}) \end{aligned}$$

Through this formula, we could find that the calculation is impossible because of the combination explosion of different POS tags. Generally, we use a n-gram especially $n = 2$ model to calculate Pr(t) approximately as follows.

$$\begin{aligned} Pr(t) = {} & Pr(t_1|t_0) \times Pr(t_2|t_1) \times Pr(t_3|t_2) \times \\ & ... \times Pr(t_T|t_{T-1}) \end{aligned}$$

where t_0 is nothing. For $Pr(w|t)$, with an independent assumption, it can be calculated approximately as follows.

$$\begin{aligned} Pr(w|t) = {} & Pr(w_1|t_1) \times Pr(w_2|t_2) \times Pr(w_3|t_3) \\ & ... \times Pr(w_T|t_T) \end{aligned}$$

Usually, the probability $Pr(t_i|t_{i-1})$ is called transition probability, and $Pr(w_i|t_i)$ is called the emission probability. They both can be estimated from the training set. This means that the tag t_i of word w_i is only determined by the tag t_{i-1} of word w_{i-1}. So, we could find the best sequence through a forward(left to right) process.

If we state all possible POS tags(stats) of each word and connect all possible t_{i-1} with all possible t_i and each edge is weighted by $Pr(t_i|t_{i-1})$, we could get a Directed Acyclic Graph(DAG). The searching process(decoding) that is involved in finding t that maximizes $Pr(t|w)$ can be explained as finding the path with the maximal probability. For this sub task, Viterbi is an efficient algorithm that can be used(Allen, 1995).

3 Lexicalized Hidden Markov Model

Lexicalized HMM is an improvement to the standard HMM. It substitutes the probability $Pr(t_i|t_{i-1})$ with $Pr(t_i|t_{i-J,i-1}, w_{i-L,i-1})$, and the probability $Pr(w_i|t_i)$ with $Pr(w_i|t_{i-K,i}, w_{i-I,i-1})$. In other words, the tag of word w_i is determined by the tags of the J words right before w_i and L words right before w_i. It uses more context information of w_i to determine its tag.

However, it will suffer from the data sparseness especially when the values of J, L, K and I are large, which means it needs an explosively larger training corpus to get a reliable estimation of these parameters, and smoothing techniques must be adopted to mitigate the problem. Back-off smoothing is used by Lexicalized HMM. In the back-off model, if a n-gram occurs more than k times in training corpus, then the estimation is used but discounted, or the estimation will use a shorter n-gram e.g. (n-1)-gram estimation as a back-off probability. So, it is a recursive process to estimate a n-gram parameter.

4 Belief Network

Belief Network is a probabilistic graphical model, which is also a DAG in which nodes represent random variables, and the arcs represent conditional independence assumptions. For example, the probability $Pr(A, B) = Pr(A) \times Pr(B|A)$ can be depicted as Figure 1(a), and if we decompose

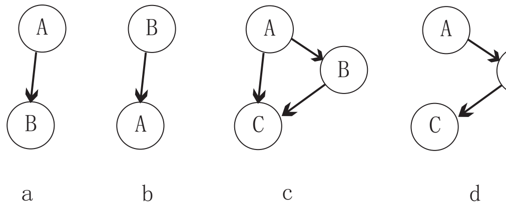


Figure 1: Some Belief Networks.

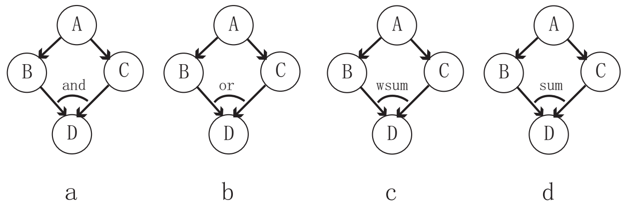


Figure 2: Canonical Belief Networks for $Pr(A, B, C, D)$.

$Pr(A, B) = Pr(B) \times Pr(A|B)$, it can be depicted as Figure 1(b). Similarly, the probability $Pr(A, B, C) = Pr(A) \times Pr(B|A) \times Pr(C|A, B)$ can be depicted as Figure 1(c).

As we have analyzed above, such decomposition would need us to estimate a large amount of parameters. In the belief network, a conditional independence relationship can be stated as follows: a node is independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes. For example, if we simplify the graph Figure 1(c) to graph Figure 1(d), it is equivalent to the decomposition: $Pr(A, B, C) = Pr(A) \times Pr(B|A) \times Pr(C|B)$, which is actually the same as that of HMM. More details about Belief Network can found in(Murphy, 2001).

5 Canonical Belief Network Based Part-Of-Speech Tagging

5.1 Canonical Belief Network

Canonical Belief Network was proposed by Turtle in 1991(Turtle, 1991), and it was used in information retrieval tasks. Four canonical forms are presented to combine different features, that is and, or, $wsum$ and sum to simplify the probability combination further. With the and relationship, it means that if a node in a DAG is true, then all of its parents must be true. With the or relationship, it means that if a node in a DAG is true, then at least one of its parents is true. With the $wsum$ relationship, it means that if a node in a DAG is true, it is determined by all of its parents and each parent has a different weight. With the sum relationship, it means that if a node in a DAG is true, it is determined by all of its parents and each parent has an equal weight.

For example, we want to evaluate the probability $Pr(D|A)$ or $Pr(D = true|A = true)$, and node D has two parents B and C, we could use the four canonical forms to evaluate $Pr(D|A)$ as shown in Figure 2. Suppose that $Pr(B|A) = p_1$ and $Pr(C|A) = p_2$, with the four canonical form and, or, $wsum$ and sum, we could get the following estimations respectively.

$$\begin{aligned} P_{and}(D|A) &= p_1 \times p_2 \\ P_{or}(D|A) &= 1 - (1 - p_1) \times (1 - p_2) \\ P_{wsum}(D|A) &= w_1 p_1 + w_2 p_2 \\ P_{sum}(D|A) &= (p_1 + p_2)/2 \end{aligned}$$

The standard Belief Network actually supposes that all the relationships are and. However, in real world, it is not the case. For example, we want to evaluate the probability that a person will use an umbrella, and there are two conditions that a person will use it: raining or a violent sunlight. If we use the standard Belief Network, it is impossible to display such situation, because it could not be raining and sunny at the same time. The or relationship could easily solve this problem.

5.2 Algorithm Description

Definition: A feature is defined as the context information of a tag/word, which can be POS tags, words or both. For example, $\{T_{i-J}, ..., T_{i-1}\}$ is a feature of tag t_i, $\{T_{i-J}, ..., T_i\}$ is a feature of word w_i, $\{T_{i-J}, ..., T_{i-1}, W_{i-L}, ..., W_{i-1}\}$ is a feature of tag t_i, $\{T_{i-K}, ..., T_i, W_{i-I}, ..., W_{i-1}\}$ is a feature of word w_i.

In our algorithm, we select 6 features for tag t_i, and select 2 features for word w_i, which are shown in Table 1. We can see that f_t^1, f_t^2 and f_t^3 are actually the n-gram features used in HMM, f_t^4, f_t^5 and f_t^6 are actually features used by lexicalized HMM.

We adopt the canonical form or to combine them as shown in Figure 3, and use the canonical form

	Features
t_i	f_t^1: $T_{i-3}, T_{i-2}, T_{i-1}$ f_t^2: T_{i-2}, T_{i-1} f_t^3: T_{i-1} f_t^4: $T_{i-3}, T_{i-2}, T_{i-1}, W_{i-3}, W_{i-2}, W_{i-1}$ f_t^5: $T_{i-2}, T_{i-1}, W_{i-2}, W_{i-1}$ f_t^6: T_{i-1}, W_{i-1}
w_i	f_w^1: T_{i-1}, T_i f_w^2: T_i

Table 1: Features used for t_i and w_i.

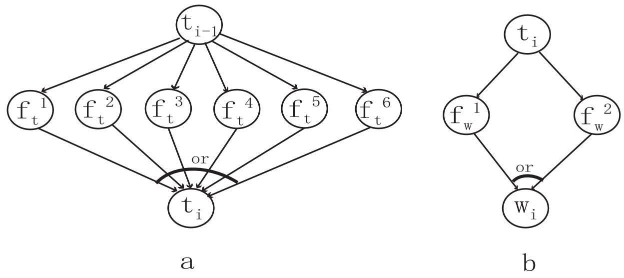


Figure 3: Canonical Belief Networks used in our algorithm.

and to combine features of t_i and w_i. Because we think that the POS tag of a new word can be determined if any one of the features can give a high confidence or implication of a certain POS tag. The probabilities $Pr(f_t^i|t_{i-1})$, $i = 1, ..., 6$. are all 1, which means that all the features in the Canonical Belief Network are considered to estimate the tag t_i of word w_i when we have already estimated the tag t_{i-1} of word w_{i-1}. So, the transition probability could be calculated as follows.

$$p_{i-1,i}^{trans} = 1 - \prod_{j=1}^{6}[1 - Pr(t_i|f_t^j)]$$

In the same way, the probabilities $Pr(f_w^i|t_i)$, $i = 1, 2$. are all 1. The emission probability could be calculated as follows.

$$p_i^{omit} = 1 - \prod_{j=1}^{2}[1 - Pr(w_i|f_w^j)]$$

Let's return to the POS tagging problem which needs to find a tag sequence t that maximizes the probability $Pr(t|w)$, given a word sequence w defined in Section 2. It is involved in evaluating two probabilities $Pr(t)$ and $Pr(w|t)$. With the Canonical Belief Network we just defined, they could be calculated as follows.

$$\begin{aligned} Pr(t) &= \textstyle\prod_{i=1}^{T} p_{i-1,i}^{trans} \\ Pr(w|t) &= \textstyle\prod_{i=1}^{T} p_i^{omit} \\ Pr(w,t) &= Pr(t) \times Pr(w|t) \end{aligned}$$

The canonical form *or* would not suffer from the data sparseness even though it refers to 4-gram, because if a 4-gram feature(f_t^1 for example) doesn't appear in the training corpus, the probability $Pr(t_i|f_t^1)$ is estimated as zero, which means the feature contributes nothing to determine the probability that word w_i gets a tag t_i, which is actually determined by a lower n-grams. Cases are the same for 3-gram, 2-gram and so on. In a special case, when a 4-gram (f_t^4 for example) appears in the training corpus and appears only once, the probability $Pr(t_i|f_t^1)$ will be 1, which means that the sentence or phrase we need to tag may have appeared in the training corpus, so we can tag the sentence or phrase with reference to the appeared sentence or phrase in the training corpus. This is an intuitional comprehension of our algorithm and its motivation.

Decoding: The problem of using high n-gram is the combination explosion especially for high grams. For example, consider the feature , suppose one word has 3 possible tags on average, then we have to evaluate $3^3 = 27$ cases for f_t^1, further, different features could get different combinations and the number of combinations will be $27^2 \times 9^2 \times 3^2 = 531441$. To solve the problem, we constrain all features to be consistent. For example, the tag t_{i-1} of feature f_t^1 must be same as that of feature f_t^2, f_t^3, f_t^4, f_t^5 and f_t^6 at one combination. The following features are not consistent, because the t_{i-1} in f_t^1 is VBP, while the t_{i-1} in f_t^4 is NN.

$f_t^1 = JJ, NNS, VBP$

$f_t^4 = JJ, NNS, NN, little, boys, book$

This will decrease the total combination to $3^3 = 27$. We use a greedy search scheme that is based on the classic decoding algorithm Viterbi. Suppose that the Viterbi algorithm has reached the state t_{i-1}, to calculate the best path from the start to t_i, we only use the tags on the best path from the start to t_{i-1} to calculate the probability. This decreases the total com-

bination to 3(the number of possible tags of t_{i-1}), which is the same as that of standard HMM.

6 Experiments

Dataset: We conduct our experiments on a Chinese corpus consisting of all news from January, 1998 of People's Daily, tagged with the tag set of Peking University(PKU), which contains 46 POS tags[1]. For the corpus, we randomly select 90% as the training set and the remaining 10% as the test set. The corpus information is shown in Table 2, where unknown words are the words that appear in test set but not in training set. The experiments are run on a machine with 2.4GHZ CPU, and 1GB memory.

	Training set	Test set
Words	1021592	112321
Sentences	163419	17777
Unknow words		2713

Table 2: Chinese corpus information.

Unknown Words: In our experiments, we first store all the words with their all possible POS tags in a dictionary. So, our algorithm gets all possible tags of a word through a dictionary. As for the word in the test set that doesn't appear in the training set, we give the probability $Pr(w_i|f_w^j)$ value 1, with all j. This processing is quite simple, however, it is enough to observe the relative performances of different POS taggers.

For Chinese word segmentation, we use the segmentation result of ICTCLAS3.0[2]. The segmentation result is shown in Table 3. *Sen-Prec* is the ratio of the sentences that are correctly segmented among all sentences in the test set.

Precision	Recall	F1	Sen-Prec
0.9811	0.9832	0.9822	0.9340

Table 3: Segmentation Result by ICTCLAS.

Open Test: We compare the POS tagging performance of our algorithm with the standard HMM, and ICTCLAS3.0. The experimental result is shown in Table 4. *Prec-Seg* is the POS tagging precision on the words that are correctly segmented. *Prec-Sen* is the ratio of the sentences that are correctly tagged among all sentences in the test set. *Prec-Sen-Seg* is the ratio of sentences that are correctly tagged among the sentences that are correctly segmented.

With the experiments, we can see that, our algorithm always gets the best performance. The ICTCLAS3.0 doesn't perform very well. However, this is probably because of that the tag set used by ICTCLAS3.0 is different from that of PKU. Even though it provides a mapping scheme from their tags to PKU tags, they may be not totally consistent. The published POS tagging precision of ICTCLAS3.0 is 94.63%, also our algorithm is a little better. This has proved that our algorithm is more effective for POS tagging task.

	ICTCLAS	HMM	CBN
Precision	0.9096	0.9388	**0.9465**
Recall	0.9115	0.9408	**0.9485**
F1	0.9105	0.9398	**0.9475**
Prec-Seg	0.9271	0.9569	**0.9647**
Prec-Sen	0.6342	0.7404	**0.7740**
Prec-Sen-Seg	0.6709	0.7927	**0.8287**

Table 4: Open test comparison result on Chinese corpus.

Close Test: As we have analyzed above in Section 5.2 that our algorithm takes advantage of more information in the training set. When a sentence or a phrase appears in the training set, it will help a lot to tag the new sentence correctly. To test whether this case really happens, we conduct a new experiment that is the same as the first one except that the test set is also added to the training set. The experimental result is shown in Table 5. We can see that the performance of our algorithm is greatly improved, while the HMM doesn't improve much, which further proves our analysis.

Even though our algorithm gives a satisfying performance, it may be able to be improved by adopting smoothing techniques to take advantage of more useful features, e.g. to make the probabilities such as $Pr(t_i|f_t^1)$, $Pr(t_i|f_t^2)$ not be zero. In addition, the adoption of techniques to deal with unknown words

[1] http://icl.pku.edu.cn/Introduction/corpustagging.htm

[2] ICTCLAS3.0 is a commercial software developed by Institute of Computing Technology, Chinese Academy of Science, that is used for Chinese word segmentation and POS tagging.

	ICTCLAS	HMM	CBN
Precision	0.9096	0.9407	**0.9658**
Recall	0.9115	0.9427	**0.9678**
F1	0.9105	0.9417	**0.9668**
Prec-Seg	0.9271	0.9588	**0.9843**
Prec-Sen	0.6342	0.7476	**0.8584**
Prec-Sen-Seg	0.6709	0.8004	**0.9191**

Table 5: Close test comparison result on Chinese corpus.

and techniques to combine with rules may also improve the performance of our algorithm. If we have a larger training corpus, it may be better to remove some confusing features such as f_t^3 and f_w^2, because they contain weak context information and this is why a higher n-gram model always performs better than a lower n-gram model when the training corpus is large enough. However, this should be validated further.

7 Conclusion and Future Work

In this paper, we present a novel algorithm that combines useful context features by Canonical Belief Network to together determine the tag of a new word. The 'or' node can allow us to use higher n-gram model although the training corpus may be not sufficient. In other words, it can overcome the data sparseness problem and make use of more information from the training corpus. We conduct experiments on a Chinese popular corpus to evaluate our algorithm, and the results have shown that it is powerful even in case that we don't deal with the unknown words and smooth the parameters.

We think that our algorithm could also be used for tagging English corpus. In addition, we only extract simple context information as features. We believe that there exists more useful features that can be used to improve our algorithm. For example, the syntax analysis could be combined as a new feature, because a POS sequence may be illegal even though it gets the maximal probability through our algorithm. Yet, these will be our future work.

Acknowledgement This work was supported by Chinese 973 Research Project under grant No. 2002CB312006.

References

Adwait Ratnaparkhi. 1996. *A Maximum Entropy Model for Part-Of-Speech Tagging*. In Proc. of the Empirical Methods in Natural Language Processing Conference(EMNLP'96), 133-142.

Bernard Merialdo. 1994. *Tagging English Text with a Probabilistic Model*. Computational Linguistics, 20(2):155–172.

Doug Cutting, Julian Kupied, Jan Pedersen and Penelope Sibun. 1992. *A Practical part-of-speech tagger*. In Proceedings of the 3rd Conference on Applied Natural Language Processing(ANLP'92), 133-140.

Eric Brill. 1992. *A simple rule-based part of speech tagger*. In Proc. of the 30th Conference on Applied Computational Linguistics(ACL'92), Trento, Italy, 112-116.

Eric Brill. 1994. *Some Advances in Transformation-Based Part of Speech Tagging*. In Proc. of the 12th National Conference on Artificial Intelligence(AAAI'94), 722-727.

Howard Turtle and W. Bruce Croft. 1991. *Evaluation of an Inference Network-Based Retrieval Model*. ACM Transactions on Information Systems, 9(3):187-222.

Hwee Tou Ng and Jin Kiat Low. 2004. *Chinese Part-of-Speech Tagging: One-at-a-Time or All-at-Once? Word-Based or Character-Based?*. In Proc. of the Empirical Methods in Natural Language Processing Conference(EMNLP'04).

James Allen. 1995. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company.

Kevin P. Murphy. 2001. *An introduction to graphical models*. Technical report, Intel Research Technical Report.

Maosong Sun and Jiayan Zou. 2001. *A critical appraisal of the research on Chinese word segmentation(In Chinese)*. Contemporary Linguistics, 3(1):22-32.

Mengqiu Wang and Yanxin Shi. 2006. *Using Part-of-Speech Reranking to Improve Chinese Word Segmentation*. In Proc. of the 5th SIGHAN Workshop on Chinese Language Processing, 205-208.

Sang-Zoo Lee, Jun-ichi Tsujii and Hae-Chang Rim. 2000. *Lexicalized Hidden Markov Models for Part-of-Speech Tagging*. In Proc. of 18th International Conference on Computational Linguistics(COLING'00), Saarbrucken, Germany, 481-487.

Scott M. Thede and Mary P. Harper. 1999. *A Second-Order Hidden Markov Model for Part-of-Speech Tagging*. In Proc. of the 37th Conference on Applied Computational Linguistics(ACL'99), 175-182.

Language Independent Text Correction using Finite State Automata

Ahmed Hassan* **Sara Noeman** **Hany Hassan**
IBM Cairo Technology Development Center
Giza, Egypt
hasanah,noemans,hanyh@eg.ibm.com

Abstract

Many natural language applications, like machine translation and information extraction, are required to operate on text with spelling errors. Those spelling mistakes have to be corrected automatically to avoid deteriorating the performance of such applications. In this work, we introduce a novel approach for automatic correction of spelling mistakes by deploying finite state automata to propose candidates corrections within a specified edit distance from the misspelled word. After choosing candidate corrections, a language model is used to assign scores the candidate corrections and choose best correction in the given context. The proposed approach is language independent and requires only a dictionary and text data for building a language model. The approach have been tested on both Arabic and English text and achieved accuracy of 89%.

1 Introduction

The problem of detecting and correcting misspelled words in text has received great attention due to its importance in several applications like text editing systems, optical character recognition systems, and morphological analysis and tagging (Roche and Schabes, 1995). Other applications, like machine translation and information extraction, operate on text that might have spelling errors. The automatic detection, and correction of spelling erros should be of great help to those applications.

The problem of detecting and correcting misspelled words in text is usually solved by checking whether a word already exists in the dictionary or not. If not, we try to extract words from the dictionary that are most similar to the word in question. Those words are reported as candidate corrections for the misspelled word.

Similarity between the misspelled word and dictionary words is measured by the Levenshtein edit distance (Levenshtein, 1966; Wagner and M.Fisher, 1974). The Levenshtein edit distance is usually calculated using a dynamic programming technique with quadratic time complexity (Wagner and M.Fisher, 1974). Hence, it is not reasonable to compare the misspelled word to each word in the dictionary while trying to find candidate corrections.

The proposed approach uses techniques from finite state theory to detect misspelled words and to generate a set of candidate corrections for each misspelled word. It also uses a language model to select the best correction from the set of candidate corrections using the context of the misspelled word. Using techniques from finite state theory, and avoiding calculating edit distances makes the approach very fast and efficient. The approach is completely language independent, and can be used with any language that has a dictionary and text data to building a language model.

The rest of this paper will proceed as follows. Section 2 will present an overview of related work. Section 3 will discuss the different aspects of the proposed approach. Section 4 presents a performance evaluation of the system. Finally a conclusion is presented in section 5.

2 Related Work

Several solutions were suggested to avoid computing the Levenshtein edit distance while finding candidate corrections. Most of those solutions select a number of dictionary words that are supposed to contain the correction, and then measure the distance between the misspelled word and all selected words. The most popular of those methods are the similarity keys methods (Kukich, 1992; Zobel and Dart, 1995; De Beuvron and Trigano, 1995). In

*Now with the University of Michigan Ann Arbor, hassanam@umich.edu

those methods, the dictionary words are divided into classes according to some word features. The input word is compared to words in classes that have similar features only.

In addition to the techniques discussed above, other techniques from finite state automata have been recently proposed. (Oflazer, 1996) suggested a method where all words in a dictionary are treated as a regular language over an alphabet of letters. All the words are represented by a finite state machine automaton. For each garbled input word, an exhaustive traversal of the dictionary automaton is initiated using a variant of Wagner-Fisher algorithm (Wagner and M.Fisher, 1974) to control the traversal of the dictionary. In this approach Levenshtein distance is calculated several times during the traversal. The method carefully traverses the dictionary such that the inspection of most of the dictionary states is avoided. (Schulz and Mihov, 2002) presents a variant of Oflazers's approach where the dictionary is also represented as deterministic finite state automaton. However, they avoid the computation of Levenshtein distance during the traversal of the dictionary automaton. In this technique, a finite state acceptor is constructed for each input word. This acceptor accepts all words that are within an edit distance k from the input word. The dictionary automaton and the Levenshtein-automaton are then traversed in parallel to extract candidate corrections for the misspelled word. The authors present an algorithm that can construct a deterministic Levenshtein-automaton for an arbitrary word of degrees 1, and 2 which corresponds to 1 or 2 errors only. They suggest another algorithm that can construct a non-deterministic Levenshtein-automaton for any other degree. They report results using a Levenshtein-automaton of degree 1(i.e. words having a single insertion, substitution, or deletion) only.

The method we propose in this work also assumes that the dictionary is represented as a deterministic finite state automaton. However, we completely avoid computing the Levenshtein-distance at any step. We also avoid reconstructing a Levenshtein-automaton for each input word. The proposed method does not impose any constraints on the bound k, where k is the edit distance between the input word and the candidate corrections. The approach can adopt several constraints on which characters can substitute certain other characters. Those constraints are obtained from a phonetic and spatial confusion matrix of characters.

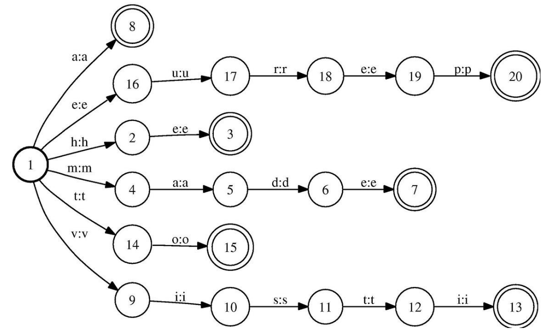


Figure 1: An FSM representation of a word list

The purpose of context-dependent error correction is to rank a set of candidate corrections taking the misspelled word context into account. A number of approaches have been proposed to tackle this problem that use insights from statistical machine learning (Golding and Roth, 1999), lexical semantics (Hirst and Budanitsky, 2005), and web crawls (Ringlstetter et al., 2007).

3 Error Detection and Correction in Text Using FSMs

The approach consists of three main phases: detecting misspelled words, generating candidate corrections for them, and ranking corrections. A detailed description of each phase is given in the following subsections.

3.1 Detecting Misspelled Words

The most direct way for detecting misspelled words is to search the dictionary for each word, and report words not found in the dictionary. However, we can make use of the finite state automaton representation of the dictionary to make this step more efficient. In the proposed method, we build a finite state machine (FSM) that contains a path for each word in the input string. This FSM is then composed with the dictionary FSM. The result of the composition is merely the intersection of the words that exist in both the input string and the dictionary. If we calculated the difference between the FSM containing all words and this FSM, we get an FSM with a path for

each misspelled word. Figure 1 illustrate an FSM that contain all words in an input string.

3.2 Generating Candidate Corrections

The task of generating candidate corrections for misspelled words can be divided into two sub tasks: Generating a list of words that have edit distance less than or equal k to the input word, and selecting a subset of those words that also exist in the dictionary. To accomplish those tasks, we create a single transducer(Levenshtein-transducer) that is when composed with an FSM representing a word, generates all words withing any edit distance k from the input word. After composing the misspelled word with Levenshtein-transducer, we compose the resulting FSM with the dictionary FSM to filter out words that do not exist in the dictionary.

3.2.1 Levenshtein-transducers for primitive edit distances

To generate a finite state automaton that contain all words within some edit distance to the input word, we use a finite state transducer that allows editing its input according to the standard Levenshtein-distance primitive operations: substitution, deletion, and insertion.

A finite-state transducers (FST) is a a 6-tuple $(Q, \Sigma_1, \Sigma_2, \sigma, i, F)$, where Q is a set of states, Σ_1 is the input alphabet, Σ_2 is the output alphabet, i is the initial state, $F \subseteq Q$ is a set of final states, and σ is a transition function (Hopcroft and Ullman, 1979; Roche and Shabes, 1997). A finite state acceptor is a special case of an FST that has the same input/output at each arc.

Figure 2 illustrates the Levenshtein-transducer for edit distance 1 over a limited set of vocabulary (a,b, and c). We can notice that we will stay in state zero as long as the output is identical to the input. On the other hand we can move from state zero, which corresponds to edit distance zero, to state one, which corresponds to edit distance one, with three different ways:

- input is mapped to a different output (input is consumed and a different symbol is emitted) which corresponds to a substitution,
- input is mapped to an epsilon (input is consumed and no output emitted) which corresponds to a deletion, and

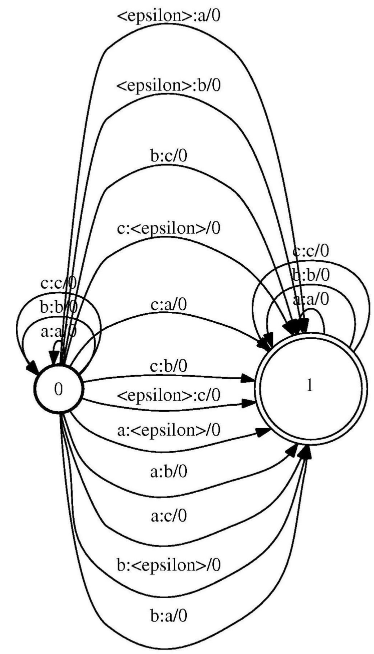


Figure 2: A Levenshtein-transducer (edit distance 1)

- an epsilon is mapped to an output (output is emitted without consuming any input) which corresponds to an insertion.

Once we reach state 1, the only possible transitions are those that consume a symbol and emit the same symbol again and hence allowing only one edit operation to take place.

When we receive a new misspelled word, we represent it with a finite state acceptor that has a single path representing the word, and then compose it with the Levenshtein-transducer. The result of the composition is a new FSM that contains all words with edit distance 1 to the input word.

3.2.2 Adding transposition

Another non-primitive edit distance operation that is frequently seen in misspelled words is transposition. Transposition is the operation of exchanging the order of two consecutive symbols ($ab \rightarrow ba$). Transposition is not a primitive operation because it can be represented by other primitive operations. However, this makes it a second degree operation. As transposition occurs frequently in misspelled words, adding it to the Levenshtein-transducer as a single editing operation would be of great help.

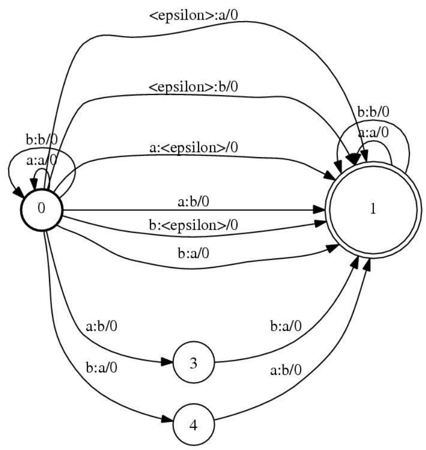


Figure 3: A Levenshtein-transducer for edit distance 1 with transposition

To add transposition, as a single editing operation, to the Levenshtein-transducer we add arcs between states zero and one that can map any symbol sequence xy to the symbol sequence yx, where x, and y are any two symbols in the vocabulary. Figure 3 shows the Levenshtein-transducer with degree 1 with transposition over a limited vocabulary (a and b).

3.2.3 Adding symbol confusion matrices

Adding a symbol confusion matrix can help reduce the number of candidate corrections. The confusion matrix determines for each symbol a set of symbols that may have substituted it in the garbled word. This matrix can be used to reduce the number of candidate corrections if incorporated into the Levenshtein-transducer. For any symbol x, we add an arc $x : y$ between states zero, and one in the transducer where $y \in Confusion_Matrix(x)$ rather than for all symbols y in the vocabulary.

The confusion matrix can help adopt the methods to different applications. For example, we can build a confusion matrix for use with optical character recognition error correction that captures errors that usually occur with OCRs. When used with a text editing system, we can use a confusion matrix that predicts the confused characters according to their phonetic similarity, and their spatial location on the keyboard.

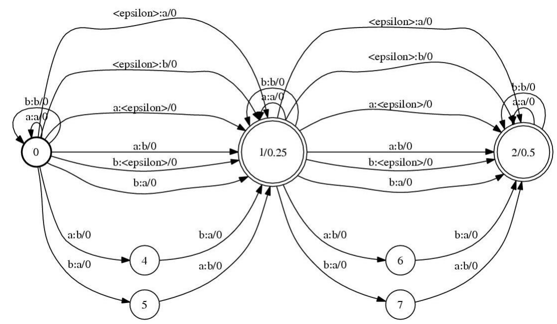


Figure 4: A Levenshtein-transducer for edit distance 2 with transposition

3.2.4 Using degrees greater than one

To create a Levenshtein-transducer that can generate all words within edit distance two of the input word, we create a new state (2) that maps to two edit operations, and repeat all arcs that moves from state 0 to state 1 to move from state 1 to state 2.

To allow the Levenshtein-transducer of degree two to produce words with edit distance 1 and 2 from the input word, we mark both state 1, and 2 as final states. We may also favor corrections with lower edit distances by assigning costs to final states, such that final states with lower number of edit operations get lower costs. A Levenshtein-transducer of degree 2 for the limited vocabulary (a and b) is shown in figure 4.

3.3 Ranking Corrections

To select the best correction from a set of candidate corrections, we use a language model to assign a probability to a sequence of words containing the corrected word. To get that word sequence, we go back to the context where the misspelled word appeared, replace the misspelled word with the candidate correction, and extract n *ngrams* containing the candidate correction word in all possible positions in the *ngram*. We then assign a score to each *ngram* using the language model, and assign a score to the candidate correction that equals the average score of all *ngrams*. Before selecting the best scoring correction, we penalize corrections that resulted from higher edit operations to favor corrections with the minimal number of editing operations.

	Edit 1/with trans.		Edit 1/no trans.		Edit 2/with trans.		Edit 2 / no trans.	
word len.	**av. time**	**av. correcs.**	**av. time**	**av. correcs.**	**av. time**	**av. correcs.**	**av. time**	**av. correcs.**
3	3.373273	18.769	2.983733	18.197	73.143538	532.637	69.709387	514.174
4	3.280419	4.797	2.796275	4.715	67.864291	136.230	66.279842	131.680
5	3.321769	1.858	2.637421	1.838	73.718353	33.434	68.695935	32.461
6	3.590046	1.283	2.877242	1.277	75.465624	11.489	69.246055	11.258
7	3.817453	1.139	2.785156	1.139	78.231015	6.373	72.2057	6.277
8	4.073228	1.063	5.593761	1.062	77.096026	4.127	73.361455	4.066
9	4.321661	1.036	3.124661	1.036	76.991945	3.122	73.058418	3.091
10	4.739503	1.020	3.2084	1.020	75.427416	2.706	72.2143	2.685
11	4.892105	1.007	3.405101	1.007	77.045616	2.287	71.293116	2.281
12	5.052191	0.993	3.505089	0.993	78.616536	1.910	75.709801	1.904
13	5.403557	0.936	3.568391	0.936	81.145124	1.575	78.732955	1.568

Table 1: Results for English

	Edit 1/with trans.		Edit 1/no trans.		Edit 2/with trans.		Edit 2 / no trans.	
word len.	**av. time**	**av. correcs.**	**av. time**	**av. correcs.**	**av. time**	**av. correcs.**	**av. time**	**av. correcs.**
3	5.710543	31.702	4.308018	30.697	83.971263	891.579	75.539547	862.495
4	6.033066	12.555	4.036479	12.196	80.481281	308.910	71.042372	296.776
5	7.060306	6.265	4.360373	6.162	79.320644	104.661	69.71572	100.428
6	9.08935	4.427	4.843784	4.359	79.878962	51.392	74.197127	48.991
7	8.469497	3.348	5.419919	3.329	82.231107	24.663	70.681298	23.781
8	10.078842	2.503	5.593761	2.492	85.32005	13.586	71.557569	13.267
9	10.127946	2.140	6.027077	2.136	83.788916	8.733	76.199034	8.645
10	11.04873	1.653	6.259901	1.653	92.671732	6.142	81.007893	6.089
11	12.060286	1.130	7.327353	1.129	94.726469	4.103	77.464609	4.084
12	13.093397	0.968	7.194902	0.967	95.35985	2.481	82.40306	2.462
13	13.925067	0.924	7.740105	0.921	106.66238	1.123	78.966914	1.109

Table 2: Results for Arabic

4 Experimental Setup

4.1 Time Performance

The proposed method was implemented in C++ on a 2GHz processor machine under Linux. We used 11,000 words of length 3,4,..., and 13, 1,000 word for each word length, that have a single error and computed correction candidates. We report both the average correction time, and the average number of corrections for each word length. The experiment was run twice on different test data, one with considering transposition as primitive operation, and the other without. We also repeated the experiments for edit distance 2 errors, and also considered the two cases where transposition is considered as a primitive operation or not. Table 1 shows the results for an English dictionary of size 225,400 entry, and Table 2 shows the results for an Arabic dictionary that has 526,492. entries.

4.2 Auto-correction accuracy

To measure the accuracy of the auto-correction process, we used a list of 556 words having common spelling errors of both edit distances 1 and 2. We put a threshold on the number of characters per word to decide whether it will be considered for edit distance 1 or 2 errors. When using a threshold of 7, the spell engine managed to correct 87% of the words. This percentage raised to 89% when all words were considered for edit distance 2 errors. The small degradation in the performance occured because in 2% of the cases, the words were checked for edit distance 1 errors although they had edit distance 2 errors. Figure 6 shows the effect of varying the characters limit on the correction accuracy.

Figure 5 shows the effect of varying the weight assigned to corrections with lower edit distances on the accuracy. As indicated in the figure, when we only consider the language model weight, we get accuracies as low as 79%. As we favor corrections with lower edit distances the correction accuracy raises, but occasionally starts to decay again when emphasis on the low edit distance is much larger than that on the language model weights.

Finally, we repeated the experiments but with us-

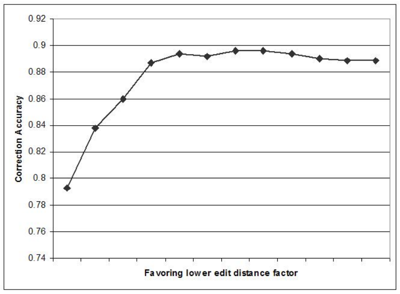


Figure 5: Effect of increasing lower edit distance favoring factor on accuracy

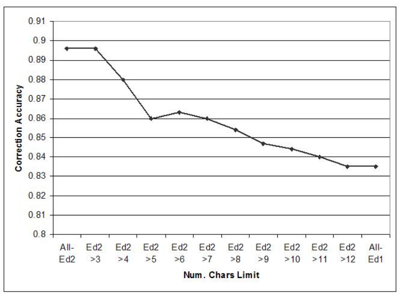


Figure 6: Effect of increasing Ed1/Ed2 char limits on accuracy

ing a confusion matrix, as 3.2.3. We found out that the average computation time dropped by 78% (below 1 ms for edit distance 1 errors) at the price of losing only 8% of the correction accuracy.

5 Conclusion

In this work, we present a finite state automata based spelling errors detection and correction method. The new method avoids calculating the edit distances at all steps of the correction process. It also avoids building a Levenshtein-automata for each input word. The method is multilingual and may work for any language for which we have an electronic dictionary, and a language model to assign probability to word sequences. The preliminary experimental results show that the new method achieves good performance for both correction time and accuracy. The experiments done in this paper can be extended in several directions. First, there is still much room for optimizing the code to make it faster especially the FST composition process. Second, we can allow further editing operations like splitting and merging.

References

Francois De Bertrand De Beuvron and Philippe Trigano. 1995. Hierarchically coded lexicon with variants. *International Journal of Pattern Recognition and Artificial Intelligence*, 9:145–165.

Andrew Golding and Dan Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine learning*, 34:107–130.

Graeme Hirst and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87–111.

J.E. Hopcroft and J.D. Ullman. 1979. Introduction to automata theory, languages, and computation. Reading, Massachusetts: Addison-Wesley.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, pages 377–439.

V.I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*.

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22:73–89.

Christoph Ringlstetter, Max Hadersbeck, Klaus U. Schulz, and Stoyan Mihov. 2007. Text correction using domain dependent bigram models from web crawls. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2007) Workshop on Analytics for Noisy Unstructured Text Data*.

Emmanuel Roche and Yves Schabes. 1995. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 2:227253.

Emmanuel Roche and Yves Shabes. 1997. Finite-state language processing. Cambridge, MA, USA: MIT Press.

Klaus Schulz and Stoyan Mihov. 2002. Fast string correction with levenshtein-automata. *International Journal of Document Analysis and Recognition (IJDAR)*, 5:67–85.

R.A. Wagner and M.Fisher. 1974. The string-to-string correction problem. *Journal of the ACM*.

Justin Zobel and Philip Dart. 1995. Finding approximate matches in large lexicons. *Software Practice and Experience*, 25:331–345.

Semantic Role Labeling of Chinese Using Transductive SVM and Semantic Heuristics

Yaodong Chen **Ting Wang** **Huowang Chen** **Xishan Xu**
Department of Computer Science and Technology, School of Computer,
National University of Defense Technology
No.137, Yanwachi Street, Changsha, Hunan 410073, P.R.China
{yaodongchen, tingwang, hwchen}@nudt.edu.cn xxs@hnmcc.com

Abstract

Semantic Role Labeling (SRL) as a Shallow Semantic Parsing causes more and more attention recently. The shortage of manually tagged data is one of main obstacles to supervised learning, which is even serious in SRL. Transductive SVM (TSVM) is a novel semi-supervised learning method special to small mount of tagged data. In this paper, we introduce an application of TSVM in Chinese SRL. To improve the performance of TSVM, some heuristics have been designed from the semantic perspective. The experiment results on Chinese Propbank showed that TSVM outperforms SVM in small tagged data, and after using heuristics, it performs further better.

1 Introduction

Semantic analysis is one of the fundamental and key problems for the research in computational linguistics. Traditional semantic research is mainly concerned with deep analysis, which provides a representation of the sentence in predicate logic or other formal specification. Recently, shallow semantic parsing is becoming a hotspot in semantic analysis research. Semantic Role Labeling is a shallow semantic parsing technology and defined as a shared task in CoNLL-04. It aims at recognizing semantic roles (i.e. arguments) for each target verb in sentence and labeling them to the corresponding syntactic constituents. Many SRL research utilizes machine learning methods (Park, 2005; Pradhan, 2005; Cohn, 2005), in which the high performance reported was attributed to large tagged dataset (Carreras, 2005). But one of the main obstacles to supervised learning is the shortage of manually labeled data, which is even serious in SRL. It could bring about one question: whether these methods perform well when large mount of tagged data are not available? In this paper, we investigate Transductive SVM (Joachims, 1999), a semi-supervised learning method, for this question. The proposed method uses large untagged data in training with the support of the linguistic knowledge of semantic roles.

Generally speaking, not all constituents in syntactic tree could act as argument candidates in SRL. Large redundant constituents lead to a high training cost and decrease the performance of statistical model especially when tagged data is small. In contrast to the pruning algorithms in Park (2005) and Xue (2004) which are based on syntax, some argument-specific heuristics, based on word semantic features of arguments, make semantic restrictions on constituent candidates to optimize dataset of statistical models. The experiment results on Chinese Propbank shows that TSVM outperforms regular statistical models in small tagged data, and after using argument-specific heuristics, it performs further better.

The rest of this paper is organized as follows. Section 2 gives the definition, method, and resources about SRL. Section 3 discusses how to apply TSVM for SRL. Some argument-specific heuristics are introduced in Section 4. And then, section 5 shows the experiment results of the proposed methods and compare it with SVM. Finally, we conclude our work in section 6.

2 Problem Definitions & Related Works

Comparing with full parsing, SRL acts on part of constituents in sentences in order to achieve high performance and robustness, as well as low complexity in practices. The SRL problem can be described as follows.

Definition Given a semantic role (or argument) collect *R* and a sentence *S*, for any substring *c* of *S*, SRL is a function: $c \rightarrow R \cup NONE$, where *NONE* is the value excluded in *R*.

Notice that *c* usually indicates phrases in a sentence. SRL can be classified to two steps:

- **Identification:** $c \rightarrow \{NONE, ARG\}$. It is a binary-value function where *ARG* is assigned to *c* when it should be labeled at some element of *R*, or *NONE* is assigned. Identification separates the argument substrings from the rest of sentence, in another words, finds the argument candidates.
- **Classification:** $c \rightarrow R$. It is a multi-value function which assigns a role value to *c*, that is, labels a role to some candidate.

Some typical systems, based on inductive learning, have been evaluated in CoNLL-05 (Carreras, 2005). It concluded that the performance of SRL depends on the combination of several factors including models, features, and results of syntactic parsing. The best result achieved F1=75.04 [1]. These systems have strong dependency on large tagged data. This paper evaluates the performance of a classical supervised learning method--SVM in small tagged data and introduces a novel semi-supervised method to handle this problem.

There are two tagged corpora available for SRL: one is Proposition Bank (Propbank); the other is FrameNet. The Propbank annotates the Penn Treebank with verb argument structure according as Levin class (Levin, 1993). It defines a general set of arguments for all types of predicates, and these arguments are divided into core and adjunct ones. FrameNet, as a linguistic ontology, describe the scenario related to each predicates. The scenario (i.e. frame) is filled with specific participants (i.e. role). In this paper, we use Chinese Propbank 1.0 provided by Linguistic Data Consortium (LDC), which is based on Chinese Treebank. It consists of 37,183 propositions indexed to the first 250k words in Chinese Treebank 5.1, including 4,865 verb types and 5,298 framesets.

[1] F1 measure computes the harmonic mean of precision and recall of SRL systems in CoNLL-2005

3 TSVM based SRL

3.1 TSVM

There are two kinds of learning modes that are applied in Artificial Intelligence, i.e. inductive inference and transductive inference. In classification problems, inductive inference trains a global model based on tagged instances from the whole problem space and classify new untagged instances by it. The classical statistical models such as SVM, ME have been developed in this way. Since large mount of tagged data are usually acquired difficultly in practice, and the global models are hard to get when tagged training data are not enough to find the target function in the hypothesis space. In addition, this global model may be unnecessary sometimes when we only care for specific data. Compared with inductive inference, transductive inference classifies untagged instances by a local model based on the clustering distribution of these untagged instances. The TSVM, a representative of transductive inference method, was introduced by Joachims (1999). TSVM is a good semi-supervised method special to some cases where the tagged data is difficult to acquire on a large scale while large untagged data is easily available. TSVM can be formulated as an optimization problem:

Minimize Over $(y_1^*..y_n^*, w, b, \xi_1..\xi_n, \xi^*..\xi_k^*)$ in

$$\frac{1}{2}\bar{w}^T\bar{w} + C\cdot\sum_{i=1}^{n}\xi_i + C^*\cdot\sum_{i=1}^{k}\xi_i^*, \text{ subject to:}$$

$$\forall_{i=1}^{n} y_i(\bar{w}\cdot\bar{x}_i + b) > 1 - \xi_i \quad \text{for } \forall_{i=1}^{n}\xi_i \geq 0$$

$$\forall_{i=1}^{k} y_i^*(\bar{w}\cdot\bar{x}_i^* + b) > 1 - \xi_i^* \quad \text{for } \forall_{i=1}^{k}\xi_i^* \geq 0$$

where $(x_1,y_1),\ldots,(x_n,y_n) \in S_{train}$, $y_1,\ldots,y_n \in \{-1,+1\}$, $x_1^*,\ldots,x_n^* \in S_{test}$, $y_1^*,\ldots,y_n^*$ is the labels of $x_1^*,\ldots, x_n^*$, C and C^*, specified by user, are the effect factor of the tagged and untagged examples respectively, $C^*\xi_i^*$ is the effect term of the *i*th untagged example in the above objective function. In addition, a cost-factor C_{temp}, which indicates the ratio of positive untagged examples, should be specified experientially by user before training.

Here we introduce the algorithm briefly, and the detail is referred to Joachims (1999). The algorithm starts with training regular SVM with the

tagged examples and then classifies the untagged examples by the trained model. Then several couples of examples (one is positive, the other is negative) are switched in class labels according to some rule, and the model is retrained to minimum the objective function. At the same time, C_{temp} will increase in consistent way. The iteration will end when C_{temp} goes beyond C^*. The algorithm is proved to converge in a finite number of steps.

3.2 Apply TSVM for SRL

The SRL using TSVM is related to following portions:

Dataset The principle of TSVM described in above section implicitly indicates the performance depends deeply on dataset (including tagged and untagged data). In particular, tagged data have an influence on original regular SVM in the first step of training, while the untagged data will affect the final performance through the iteration of training. It is obvious that the more even the data set distribution is, the better the learning classifier will perform. Similar to most practical classification task, a serious uneven problem (Li, 2003) exists in SRL. For instance, the number of constituents labeled to arguments (positive instances) is much less than the number of the rest (negative instances). To handle this problem, we design some heuristics for several kinds of arguments (that is, ARG0, ARGM-TMP, ARGM-LOC, ARGM-MNR, ARGM-DIR and ARGM-EXT) semantically. These heuristics filter out redundant constituents and raise the ratio of positive instances in the dataset. We will compare these argument-specific heuristics with Xue (2004), and some results are showed in Section 4.

Parameters The ratio of positive examples in dataset, *P*, is a key parameter in TSVM and should be assigned as one prior value in experiment. In this paper, *P* is dynamically assigned according to different argument since different heuristics could produce different proportion of positive and negative instances used to training data.

Features A wide range of features have been shown to be useful in previous work on SRL (Pradhan, 2005; Xue et al, 2004). This paper chooses 10 features in classification because of two reasons: at first, they are the core features considered to have significance on the performance of SRL (Carreras, 2005); secondly, these features provide a standard to evaluate different methods of Chinese SRL. These features are listed in Table 1, detail description referred in Xue (2005).

Feature	Description
Predicate	The predicate lemma
Subcat-Frame	The rule that expands the parent of verb
Path	The syntactic path through the parse tree from the parse constituent to the predicate being classified
Position	A binary feature identifying whether the phrase is before or after the predicate
Phrase Type	The syntactic category of the phrase corresponding to the argument
Phrase type of the sibling to the left	The syntactic category of the phrase is sibling to the argument in the left
Head Word and Part Of Speech	The syntactic head of the phrase
First and last word of the constituent in focus	First and last word of phrase corresponding to the argument
Syntactic Frame	The syntactic frame consists of the NPs that surround the predicate

Table 1. The features of Semantic Role Labeling

It should be mentioned that we have not considered the Combination features (Xue et al, 2005) because the above 10 features have already coded them. Verb class is also not be used here since we have no idea about the syntactic alternations used for verb classification in Xue (2005) and could not evaluate them equally. So, the experiment in this paper refers to the results without verb class in Xue (2005).

Classifiers Chinese Propbank has 22 argument types, in which 7 argument types appearing less than ten times or even having no appearance have not been considered, that is,ARGM-FRQ, ARGM-ASP, ARGM-PRD, ARGM-CRD, ARGM-T, and ARGM-DGR. So we have developed 15 binary classifiers for those 15 type of arguments and excluded the above 7 because they hardly provide useful information for classification, as well as have slightly influence on results (account for 0.02% in all arguments appeared in the corpus).

4 Heuristics

In this section, we discuss the principle of the designing of the argument-specific heuristics. To handle the uneven problem in SRL, six semantic heuristics have been designed for six types of arguments, such as ARG0, ARGM-TMP, ARGM-

LOC, ARGM-MNR, ARGM-DIR, and ARGM-EXT. The heuristic is actually some restrictive rules which can be viewed as pre-processing of identification. (Xue et al, 2004) introduced a primary algorithm for pruning argument non-candidates. The algorithm still remain large redundant unnecessary constituents yet (correct arguments account for 7.31% in all argument candidates extracted). (Park, 2005) used the clause boundary restriction and tree distance restriction for extracting candidates based on Government and Binding Theory. All of these restrictive rules, however, are on the syntax level. Here we consider several semantic features directly extracted by the head word of the argument in lexicon. This is based on facts that ARG0 contain mostly NPs whose head words are animate objects or entities. (Yi, 2007) shows *agent* and *experiencer* as ARG0 accounts for 93% in all ARG0s in Propbank. In addition, some head words of the constituents labeled by ARGM-TMP have temporal sense, which is the same as ARGM-LOC whose head words usually have spatial sense. The semantic information can be extracted from a Chinese-English bilingual semantic resource: HowNet (Dong, 2000). HowNet is an on-line common-sense knowledge base providing a universal lexical concept representation mechanism. Word sense representations are encoded by a set of approximately 2,000 primitive concepts, called sememes. A word sense is defined by its primary sememes. For example, *小孩*(*child*) is defined with sememes “human|人”, “young|幼”; *目前 (at present*) has sememes “time|时间”, “now|今”; *街* (*street*) contains sememes “location|位置”, “route|路”. We considered sememes as the basis of heuristics, and Table 2 shows these heuristics.

Table 2 shows the argument-specific heuristics on the semantics level, for example, only when the head word of a PP contains a sememe “time|时间”, it could be a candidate of ARGM-TMP, such as *目前*, *当今*; only a sememe “location|位置” has a head word of one phrase, it may be labeled to ARGM-LOC. Furthermore, we make a comparison with Xue (2004) in whole argument types on Chinese Propbank (the extraction principle about argument_types which are not listed in Table 1 is the same as Xue (2004)). We find the argument-specific heuristics decrease in uneven problem more effectively than Xue (2004). The overall coverage [2] rises from 7.31% to 20.30%, that is, 65% constituents which have no possibility to labeling have been pruned based on six types of arguments. And the overall recall of arguments in corpus decline slightly from 99.36% to 97.28%.

Args	Def	Heuristic	Cover -age
ARG0	agent,ex -eriencer	the NP whose head word has sememe that is hyponymy with animate|生物 or whose head word is place or organization	38.90
ARGM-TMP	temporal	The NP and LCP whose head word has sememe time|时间 or the PP whose prep is from|从, from|自, to|到, in|於, or at|在	58.7
ARGM-LOC	location	The NP and LCP whose head word has sememe location|位置 or the PP whose prep is in|在 ,at|在 or from|于	44.4
ARGM-MNR	manner	The PP whose prep is “according to|根据, 按,据, 按照” or by|通过, as|随着	30.98
ARGM-DIR	directional	The PP whose prep is to|对 or from|从, to|向	20.56
ARGM-EXT	extent	The NP and QP whose head word is number	70.27

Table 2. The arguments-specific heuristics.

5 Experiment and discussion

This section will describe the experiment on the SRL in Chinese Treebank, compare TSVM with regular SVM, and evaluate the effect of the proposed argument-specific heuristics.

5.1 Experiment Setting

SVM-light[3] is used as a SVM classifier toolkit in the experiment, which includes some sub-tools for optimizing performance and reducing training time. It also provides an approximate implementation of transductive SVM. At first, about 80% propositions (1711891) has been extracted randomly from the corpus as the dataset, which had been divided into tagged set and untagged set according to 4:1. Then, for each type of arguments,

[2] The coverage means the ratio of arguments in all role candidates extracted from Chinese Propbank by given heuristic.

[3] http://svmlight.joachims.org/

numeric vectors are extracted from these two sets (one proposition could produce many instances) as the dataset for the following learning models through the heuristics in Table 2. When training the classifier, linear kernel function had used, setting the C to 2 experientially.

5.2 Results and Discussion

A baseline was developed with 10 features and 15 SVM classifiers (tagged set for training, untagged set for testing) as described in Section 3. We made a comparison between the baseline and the work in Xue (2005), and then used the argument-specific heuristics for baseline. Table 3 shows the performance of these methods. Baseline matches Xue approximately despite of the absence of combination features. We also find that the argument-specific heuristics improve the performance of baseline from 89.97% to 90.86% for F1 and beyond the Xue. It can be explained that when using heuristics, the proportion of positive and negative instances in dataset are adjusted reasonably to improve the model. About 1 percent improvement attributes to the effectivity of these six argument-specific heuristics.

Systems	Precision	Recall	F1
Baseline	89.70	90.24	89.97
Xue	90.40	90.30	90.30
Heuristics	91.45	90.28	90.86

Table 3. A comparison among baseline, Xue and heuristics through regular SVM

In order to investigating the learning performance of SVM, TSVM and TSVM using argument-specific heuristics in small tagged data, we extracted randomly different number of propositions in Propbank as tagged data and another 5000 propositions held out as untagged data. Both of them are used for training TSVM model. Table 4 shows the overall performance and the performances of two arguments--ARG0 and ARGM-TMP--along with the different training data size. As we can see in (a) of Table 4, the TSVM leads to an improved performance on overall argument types when tagged data less than 100 propositions (raising F1 about 10%). It indicates that transductive inference performs much better than inductive inference because it makes use of the additional information about the distribution of 5000 untagged propositions. More important, we find that TSVM using argument-specific heuristics, comparing to TSVM, has a distinctive improvement (raising about 3%). It confirmed that our heuristics have positive influences on transductive inference.

Number of tagged propositions	SVM	TSVM	TSVM + Heuristics
10	36.51	50.51	50.82
20	41.65	50.52	53.66
40	41.64	55.42	60.63
160	76.40	80.84	82.32
1000	82.00	83.87	84.00
5000	84.41	85.61	86.45

(a). The overall results on all argument types.

Number of tagged propositions	SVM	TSVM	TSVM + Heuristics
10	20.51	29.51	30.21
20	22.34	32.45	38.54
40	35.00	45.42	50.63
160	45.45	50.45	55.74
1000	52.43	55.43	57.40
5000	58.00	60.34	61.45

(b) The detail results on ARG0

Number of tagged propositions	SVM	TSVM	TSVM + Heuristics
10	15.98	20.45	19.98
20	25.34	29.45	35.43
40	30.32	32.80	39.43
160	38.31	40.00	45.09
1000	48.43	50.43	55.45
5000	60.34	62.34	63.90

(c) The detail results on ARGM-TMP

Table 4. A comparison with Regular SVM, TSVM and TSVM using argument-specific heuristics holding 5000 untagged propositions

Number of untagged propositions	SVM	TSVM	TSVM + Heuristics
500	69.03	68.50	69.44
1000	70.12	70.22	70.82
2000	68.64	71.30	73.01
4000	69.53	72.01	76.50
5000	68.95	72.54	77.21
10000	70.28	74.78	79.74

Table 5. A comparison with Regular SVM, TSVM and TSVM using argument-specific heuristics holding 100 tagged propositions

We then evaluate the six argument-specific heuristics introduced in Section 4 with the same 5000 untagged propositions. It is noticeable that the training time of TSVM doubles that of SVM approximately. The (b) and (c) of Table 4 give the detail results on ARG0 and ARGM-TMP. Com-

pared with (a), it is obvious that the improvement between TSVM using heuristics with TSVM for ARG0 and ARGM-TMP is larger than the overall improvement. That is to say, the more distinctive knowledge is embedded in heuristics, the better performance can be achieved for the corresponding argument. This observation encourages us to investigate more heuristics for more arguments.

Finally, the influence of untagged data on performance of TSVM has been investigated. We extract different size of untagged propositions and hold 100 tagged propositions for training TSVM. Table 5 shows the results. It should be mention that the result of SVM fluctuates slightly, which is due to different number of testing examples. On the other hand, TSVM and TSVM using argument-specific heuristics improve highly as the increase in untagged data size. The bigger the untagged data, the larger the performance gap between SVM and TSVM and the gap between TSVM and TSVM using argument-specific heuristics. It indicates that the argument-specific heuristics, optimizing the dataset, have substantial effectivity in the performance of TSVM when untagged data is large.

6 Conclusions

Most machine learning methods such as SVM, ME have a strong dependence on tagged data, which lead to a poor generalization when large tagged data are not available. This paper introduces a novel semi-supervised method--TSVM for this problem. TSVM can effectively use clustering information from untagged data for training the model. The experiment demonstrated the TSVM achieve better performance than regular SVM when only very few tagged examples are available. Aiming at serious uneven problem in SRL, argument-specific heuristics are proposed correspond to six kinds of arguments. These heuristics are developed by extracting semantic features of arguments from HowNet. The experiment proves that these heuristics have much effect not only in the inductive inference (regular SVM) but also in transductive inference (TSVM), especially when the untagged data is large. The high performance of six heuristics demonstrated that semantic characteristics are significant on SRL, which encourages us to develop more semantic characteristics of more arguments in the future.

Acknowledgement This research is supported by the National Natural Science Foundation of China (60403050), Program for New Century Excellent Talents in University (NCET-06-0926) and the National Grand Fundamental Research Program of China under Grant (2005CB321802).

References

Levin Beth. 1993. *English Verb Class and Alternations: A Preliminary Investigation.* Chicago: University of Chicago Press.

Xavier Carreras and Llu´ıs M`arquez, 2005. *Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling*. CoNLL-2005.

Trevor Cohn and Philip Blunsom. 2005. *Semantic role labeling with tree conditional random fields*. CoNLL-2005.

Zhendong Dong. 2000. http://www.keenage.com/.

Thorsten Joachims. 1999. *Transductive inference for text classification using support vector machines.* ICML-99, pages 200–209, Bled, Slovenia, Jun.

Yaoyong Li and John Shawe-Taylor. 2003. *The SVM with Uneven Margins and Chinese Document Categorization*. PACLIC-2003, Singapore.

Kyung-Mi Park and Hae-Chang Rim. 2005. *Maximum entropy based semantic role labeling.* CoNLL-2005.

Pradhan S, Hacioglu K, Krugler V, et al. 2005. *Support Vector Learning for Semantic Argument Classification.* Machine Learning journal. 60(1-3): 11-39.

Nianwen Xue and Martha Palmer. 2004. *Calibrating Features for Semantic Role Labeling.* EMNLP.

Nianwen Xue and Martha Palmer. 2005, *Automatic Semantic Role Labeling for Chinese Verbs.* The IJCAI-2005, Edinburgh, Scotland.

Szuting Yi, Edward Loper and Martha Palmer. 2007. *Can Semantic Roles Generalize Across Genres?* NAANL-HLT 07, Rochester, N Y.

A Comparative Study of Mixture Models for Automatic Topic Segmentation of Multiparty Dialogues

Maria Georgescul
ISSCO/TIM, ETI
University of Geneva
maria.georgescul@eti.unige.ch

Alexander Clark
Department of Computer Science
Royal Holloway University of London
alexc@cs.rhul.ac.uk

Susan Armstrong
ISSCO/TIM, ETI
University of Geneva
susan.armstrong@issco.unige.ch

Abstract

In this article we address the task of automatic text structuring into linear and non-overlapping thematic episodes at a coarse level of granularity. In particular, we deal with topic segmentation on multi-party meeting recording transcripts, which pose specific challenges for topic segmentation models. We present a comparative study of two probabilistic mixture models. Based on lexical features, we use these models in parallel in order to generate a low dimensional input representation for topic segmentation. Our experiments demonstrate that in this manner important information is captured from the data through less features.

1 Introduction

Some of the earliest research related to the problem of text segmentation into thematic episodes used the word distribution as an intrinsic feature of texts (Morris and Hirst, 1991). The studies of (Reynar, 1994; Hearst, 1997; Choi, 2000) continued in this vein. While having quite different emphasis at different levels of detail (basically from the point of view of the employed term weighting and/or the adopted inter-block similarity measure), these studies analyzed the word distribution inside the texts through the instrumentality of merely one feature, i.e. the one-dimensional inter-block similarity.

More recent work use techniques from graph theory (Malioutov and Barzilay, 2006) and machine learning (Galley et al., 2003; Georgescul et al., 2006; Purver et al., 2006) in order to find patterns in vocabulary use.

We investigate new approaches for topic segmentation on corpora containing multi-party dialogues, which currently represents a relatively less explored domain. Compared to other types of audio content (e.g. broadcast news recordings), meeting recordings are less structured, often exhibiting a high degree of participants spontaneity and there may be overlap in finishing one topic while introducing another. Moreover while ending the discussion on a certain topic, there can be numerous new attempts to introduce a new topic before it becomes the focus of the dialogue. Therefore, the task of automatic topic segmentation of meeting recordings is more difficult and requires a more refined analysis. (Galley et al., 2003; Georgescul et al., 2007) dealt with the problem of topic segmentation of multiparty dialogues by combining various features based on cue phrases, syntactic and prosodic information. In this article, our investigation is based on using merely lexical features.

We study mixture models in order to group the words co-occurring in texts into a small number of semantic concepts in an automatic unsupervised way. The intuition behind these models is that a text document has an underlying structure of "latent" topics, which is hidden. In order to reveal these latent topics, the basic assumption made is that words related to a semantic concept tend to occur in the proximity of each other. The notion of proximity between semantically related words can vary for various tasks. For instance, bigrams can be considered to capture correlation between words at a very

short distance. At the other extreme, in the domain of document classification, it is often assumed that the whole document is concerned with one specific topic and in this sense all words in a document are considered to be semantically related. We consider for our application that words occurring in the same thematic episode are semantically related.

In the following, the major issues we will discuss include the formulations of two probabilistic mixture approaches, their methodology, aspects of their implementation and the results obtained when applied in the topic segmentation context. Section 2 presents our approach on using probabilistic mixture models for topic segmentation and shows comparisons between these techniques. In Section 3 we discuss our empirical evaluation of these models for topic segmentation. Finally, some conclusions are drawn in Section 4.

2 Probabilistic Mixture Models

The probabilistic latent models described in the following exploit hierarchical Bayesian frameworks. Based on prior distributions of word rate variability acquired from a training corpus, we will compute a density function to further analyze the text content in order to perform topic segmentation at a coarse level of granularity. In this model, we will be working with 'blocks' of text which consist of a fixed number of consecutive utterances.

In the following two subsections, we use the following notation:

- We consider a text corpus $\mathcal{B} = \{b_1, b_2, ..., b_M\}$ containing M blocks of text with words from a vocabulary $\mathcal{W} = \{w_1, w_2, ..., w_N\}$. M is a constant scalar representing the number of blocks of text. N is a constant scalar representing the number of terms in vocabulary $\mathcal{W}$.

- We pre-process the data by eliminating content free words such as articles, prepositions and auxiliary verbs. Then, we proceed by lemmatizing the remaining words and by adopting a bag-of-words representation. Next, we summarize the data in a matrix $\mathcal{F} = (f(b_i, w_{i,j}))_{(i,j) \in M \times N}$, where $f(b_i, w_{i,j})$ denotes the $log.entropy$ weighted frequency of word $w_{i,j}$ in block b_i.

- Each occurrence of a word in a block of text is considered as representing an observation $(w_{m,n}, b_m)$, i.e. a realization from an underlying sequence of random variables $(W_{m,n}, B_m)^{1 \leq m \leq M}_{1 \leq n \leq N}$. $w_{m,n}$ denotes the term indicator for the n-th word in the m-th block of text.

- Each pair $(w_{m,n}, b_m)$ is associated with a discrete hidden random variable $Z_{m,n}$ over some finite set $\mathcal{Z} = \{z_1, z_2, ..., z_K\}$. K is a constant scalar representing the number of mixture components to generate.

- We denote by $P(z_{m,n} = z_k)$ or simply by $P(z_k)$ the probability that the k-th topic has been sampled for the n-th word in the m-th block of text.

2.1 Aspect Model for Dyadic Data (AMDD)

In this section we describe how we apply latent modeling for dyadic data (Hofmann, 2001) to text representation for topic segmentation.

2.1.1 Model Setting

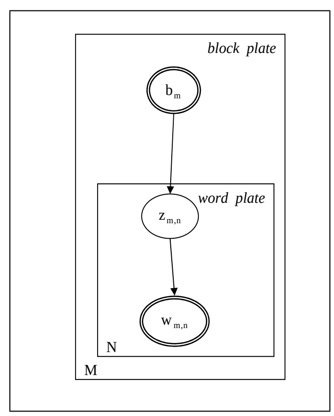


Figure 1: Graphical model representation of the aspect model.

We express the joint or conditional probability of words and blocks of text, by assuming that the choice of a word during the generation of a block of text is independent of the block itself, given some (unobserved) hidden variable, also called *latent* variable or *aspect*.

The graphical representation of the AMDD data generation process is illustrated in Figure 1 by using

the plate notation. That is, the ovals (i.e. the nodes of the graph) represent probabilistic variables. The double ovals around the variables $w_{m,n}$ and b_m denote observed variables. $z_{m,n}$ is the mixture indicator, the hidden variable, that chooses the topic for the n-th word in the m-th block of text. Arrows indicate conditional dependencies between variables. For instance, the $w_{m,n}$ variable in the word space and the b_m variable in the block space have no direct dependencies, i.e. it is assumed that the choice of words in the generation of a block of text is independent of the block given a hidden variable. The boxes represent "plates", i.e. replicates of sampling steps with the variable in the lower left corner referring to the number of samples. For instance, the "word plate" in Figure 1 illustrates N independently and identically distributed repeated trials of the random variable $w_{m,n}$.

According to the topology of the asymmetric AMDD Bayesian network from Figure 1, we can specify the joint distribution of a word $w_{m,n}$, a latent topic z_k and a block of text b_m: $P(w_{m,n}, z_k, b_m) = P(b_m) \cdot P(z_k|b_m) \cdot P(w_{m,n}|z_k)$. The joint distribution of a block of text b_m and a word $w_{m,n}$ is thus:

$$P(b_m, w_{m,n}) = \sum_{k=1}^{K} P(w_{m,n}, z_k, b_m) = P(b_m) \cdot \sum_{k=1}^{K} \underbrace{P(z_k|b_m)}_{mixing\ proportions} \cdot \underbrace{P(w_{m,n}|z_k)}_{mixture\ components} \quad (1)$$

Equation 1 describes a special case of a finite mixture model, i.e. it uses a convex combination of a set of component distributions to model the observed data. That is, each word in a block of text is seen as a sample from a mixture model, where mixture components are multinomials $P(w_{m,n}|z_k)$ and the mixing proportions are $P(z_k|b_m)$.

2.1.2 Inferring and Employing the AMDD Model

The *Expectation-Maximization (EM)* algorithm is the most popular method to estimate the parameters for mixture models to fit a training corpus. The EM algorithm for AMDD is based on iteratively maximizing the log-likelihood function: $\mathcal{L}_{PLSA} = \sum_{m=1}^{M} \sum_{n=1}^{N} f(b_m, w_{m,n}) \cdot logP(w_{m,n}, b_m)$. However, the EM algorithm for AMDD is prone to overfitting since the number of parameters to be estimated grows linearly with the number of blocks of text. In order to avoid this problem, we employed the tempered version of the EM algorithm that has been proposed by Hofmann (2001).

We use the density estimation method in AMDD to reduce the dimension of the blocks-by-words space. Thus, instead of using the words as basic units for each block of text representation, we employ a "topic" basis, assuming that a few topics will capture more information than the entire huge amount of words in the vocabulary. Thus, the m-th block of text is represented by the vector $(P(z_1|b_m), P(z_2|b_m), ..., P(z_k|b_m))$. Then, we use these posterior probabilities as a threshold to identify the boundaries of thematic episodes via support vector classification (Georgescul et al., 2006). That is, we consider the topic segmentation task as a binary-classification problem, where each utterance should be classified as marking the presence or the absence of a topic shift in the dialogue.

2.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (Blei et al., 2003) can be seen as an extension of AMDD by defining a probabilistic mixture model that includes Dirichlet-distributed priors over the masses of the multinomials $P(w|z)$ and $P(z|b)$.

2.2.1 Model Setting

In order to describe the formal setting of LDA in our context, we use the following notation in addition to those given at the beginning of Section 2:

- $\vec{\theta}_m$ is a parameter notation for $P(z|b = b_m)$, the topic mixture proportion for the m-th block of text;
- $\vec{\alpha}$ is a hyperparameter (a vector of dimension K) on the mixing proportions $\vec{\theta}_m$;
- $\Theta = \left\{\vec{\theta}_m\right\}_{m=1}^{M}$ is a matrix (of dimension $M \times K$), composed by placing the vectors $\vec{\theta}_1, \vec{\theta}_2, ..., \vec{\theta}_M$ as column components;
- $\vec{\varphi}_k$ is a parameter notation for $P(w|z_k)$, the mixture component for topic k;
- $\vec{\beta}$ is a hyperparameter (a vector of dimension N) on the mixture components $\vec{\varphi}_k$;

- $\Phi = \{\vec{\varphi}_k\}_{k=1}^{K}$ is a matrix of dimension $K \times N$ composed by placing the vectors $\vec{\varphi}_1, \vec{\varphi}_2, ..., \vec{\varphi}_K$ as column components;
- N_m denotes the length of the m-th block of text and is modeled with a Poisson distribution with constant parameter ξ;

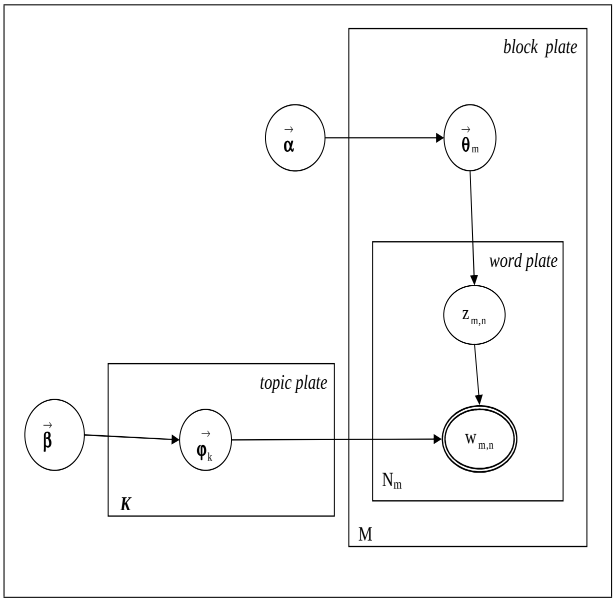


Figure 2: Graphical model representation of latent Dirichlet allocation.

LDA generates a stream of observable words $w_{m,n}$ partitioned into blocks of text $\vec{b}_m$ as shown by the graphical model in Figure 2. The Bayesian network can be interpreted as follows: the variables Φ, θ and z are the three sets of latent variables that we would like to infer. The plate surrounding $\vec{\varphi}_k$ illustrates the repeated sampling of word distributions for each topic z_k until K topics have been generated. The plate surrounding $\vec{\theta}_m$ illustrates the sampling of a distribution over topics for each block b for a total of M blocks of text. The inner plate over $z_{m,n}$ and $w_{m,n}$ illustrates the repeated sampling of topics and words until N_m words have been generated for a block $\vec{b}_m$.

Each block of text is first generated by drawing a topic proportion $\vec{\theta}_m$, i.e. by picking a distribution over topics from a Dirichlet distribution. For each word $w_{m,n}$ from a block of text $\vec{b}_m$, a topic indicator k is sampled for $z_{m,n}$ according to the block-specific mixture proportion $\vec{\theta}_m$. That is, $\vec{\theta}_m$ determines $P(z_{m,n})$. The topic probabilities $\vec{\varphi}_k$ are also sampled from a Dirichlet distribution. The words in each block of text are then generated by using the corresponding topic-specific term distribution $\vec{\varphi}_{z_{m,n}}$.

Given the graphical representation of LDA illustrated in Figure 2, we can write the joint distribution of a word $w_{m,n}$ and a topic z_k as:
$P(w_{m,n}, z_k|\vec{\theta}_m, \Phi) = P(z_k|\vec{\theta}_m) \cdot P(w_{m,n}|\vec{\varphi}_k)$.
Summing over k, we obtain the marginal distribution:

$$P(w_{m,n}|\vec{\theta}_m, \Phi) = \sum_{k=1}^{K} \left(\underbrace{P(z_k|\vec{\theta}_m)}_{mixture\ proportion} \cdot \underbrace{P(w_{m,n}|\vec{\varphi}_k)}_{mixture\ component} \right).$$

Hence, similarly to AMDD (see Equation 1), the LDA model assumes that a word $w_{m,n}$ is generated from a random mixture over topics. Topic probabilities are conditioned on the block of text a word belongs to. Moreover LDA leaves flexibility to assign a different topic to every observed word and a different proportion of topics for every block of text.

The joint distribution of a block of text $\vec{b}_m$ and the latent variables of the model $\vec{z}_m$, $\vec{\theta}_m$, Φ, given the hyperparameters $\vec{\alpha}$, $\vec{\beta}$ is further specified by:

$$P(\vec{b}_m, \vec{z}_m, \vec{\theta}_m, \Phi|\vec{\alpha}, \vec{\beta}) = \overbrace{P(\Phi|\vec{\beta})}^{topic\ plate} \cdot \underbrace{P(\vec{\theta}_m|\vec{\alpha}) \cdot \prod_{n=1}^{N_m} \overbrace{P(z_{m,n}|\vec{\theta}_m) \cdot P(w_{m,n}|\vec{\varphi}_{z_{m,n}})}^{word\ plate}}_{block\ plate}.$$

Therefore, the likelihood of a block $\vec{b}_m$ is derived as the marginal distribution obtained by summing over the $z_{m,n}$ and integrating out the distributions $\vec{\theta}_m$ and Φ.

2.2.2 Inferring and Employing the LDA Model

Since the integral involved in computing the likelihood of a block $\vec{b}_m$ is computationally intractable, several methods for approximating this posterior have been proposed, including variational expectation maximization (Blei et al., 2003) and Markov chain Monte Carlo methods (Griffiths and Steyvers, 2004).

We follow an approach based on Gibbs sampling as proposed in (Griffiths and Steyvers, 2004). As the convergence criteria for the Markov chain, we

check how well the parameters cluster semantically related blocks of text in a training corpus and then we use these values as estimates for comparable settings.

The LDA model provides a soft clustering of the blocks of text, by associating them to topics. We exploit this clustering information, by using the distribution of topics over blocks of text to further measure the inter-blocks similarity. As in Section 2.1.2, the last step of our system consists in employing binary support vector classification to identify the boundaries of thematic episodes in the text. That is, we consider as input features for support vector learning the component values of the vector $(\theta_{m,z_1}, \theta_{m,z_2}, ..., \theta_{m,z_k})$.

3 Experiments

In order to evaluate the performance of AMDD and LDA for our task of topic segmentation, in our experiments we used the transcripts of ICSI-MR corpus (Janin et al., 2004), which consists of 75 meeting recordings. A subset of 25 meetings, which are transcribed by humans and annotated with thematic boundaries (Galley et al., 2003), has been kept for testing purposes and support vector machine training. The transcripts of the remaining 50 meetings have been used for the unsupervised inference of our latent models. The fitting phase of the mixture models rely on the same data set that have been pre-processed by tokenization, elimination of stopwords and lemmatization.

Once the models' parameters are learned, the input data representation is projected into the lower dimension latent semantic space. The evaluation phase consists in checking the performance of each model for predicting thematic boundaries. That is, we check the performance of the models for predicting thematic boundaries on the same test set. The size of a block of text during the testing phase has been set to one, i.e. each utterance has been considered as a block of text.

Figure 3 compares the performance obtained for various k values, i.e. various dimensions of the latent semantic space, or equivalently different numbers of latent topics. We have chosen k=$\{50, ...400\}$ using incremental steps of 50.

The performance of each latent model is measured by the accuracy $Acc = 1 - P_k$, where P_k denotes the error measure proposed by (Beeferman et al., 1999). Note that the P_k error allows for a slight variation in where the hypothesized thematic boundaries are placed. That is, wrong hypothesized thematic boundaries occurring in the proximity of a reference boundary (i.e. in a fixed-size interval of text) are tolerated. As proposed by (Beeferman et al., 1999), we set up the size of this interval to half of the average number of words per segment in the gold standard segmentation.

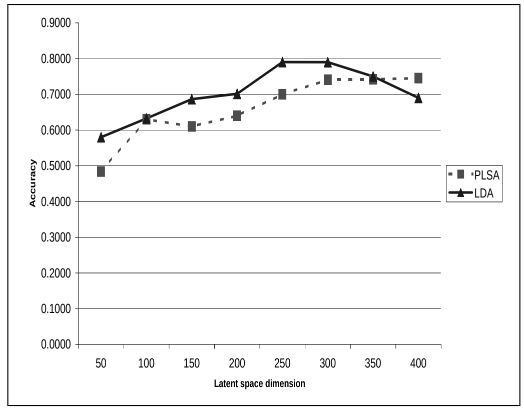


Figure 3: Results of applying the mixture models for topic segmentation.

As we observe from Figure 3, LDA and AMDD achieved rather comparable thematic segmentation accuracy. While LDA steadily outperformed AMDD, the results do not show a notable advantage of LDA over AMDD. In contrast, AMDD has better performances for less dimensionality reduction. That is, the LDA performance curve goes down when the number of latent topics exceeds over 300.

	LDA	*LCSeg*	*SVMs*
P_k error rate	21%	32 %	22%

Table 1: Comparative performance results.

In Table 1, we provide the best results obtained on ICSI data via LDA modeling. We also reproduce the results reported on in the literature by (Galley et al., 2003) and (Georgescul et al., 2006), when the evaluation of their systems was also done on ICSI data. The *LCSeg* system proposed by (Galley et al., 2003) is based on exploiting merely lexical features. Improved performance results have

been obtained by (Galley et al., 2003) when extra non-lexical features have been adopted in a decision tree classifier. The system proposed by (Georgescul et al., 2006) is based on support vector machines (SVMs) and is labeled in the table as *SVMs*. We observe from the table that our approach based on combining LDA modeling with SVM classification outperforms *LCSeg* and performs comparably to the system of Georgescul et al. (2006). Thus, our experiments show that the LDA word density estimation approach does capture important information from the data through 90% less features than a bag-of-words representation.

4 Conclusions

With the goal of performing linear topic segmentation by exploiting word distributions in the input text, the focus of this article was on both comparing theoretical aspects and experimental results of two probabilistic mixture models. The algorithms are applied to a meeting transcription data set and are found to provide an appropriate method for reducing the size of the data representation, by performing comparably to previous state-of-the-art methods for topic segmentation.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning*, 34:177–210. Special Issue on Natural Language Learning.

David M. Blei, Andrew Y. Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, pages 993–1022.

Freddy Choi. 2000. Advances in Domain Independent Linear Text Segmentation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Seattle, USA.

Michael Galley, Kathleen McKeown, Eric Fosler-Luissier, and Hongyan Jing. 2003. Discourse Segmentation of Multi-Party Conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 562–569, Sapporo, Japan.

Maria Georgescul, Alexander Clark, and Susan Armstrong. 2006. Word Distributions for Thematic Segmentation in a Support Vector Machine Approach. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 101–108, New York City, USA.

Maria Georgescul, Alexander Clark, and Susan Armstrong. 2007. Exploiting Structural Meeting-Specific Features for Topic Segmentation. In *Actes de la 14ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN)*, pages 15–24, Toulouse, France.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding Scientific Topics. In *Proceedings of the National Academy of Sciences*, volume 101, pages 5228–5235.

Marti Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64.

Thomas Hofmann. 2001. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42:177–196.

Adam Janin, Jeremy Ang, Sonali Bhagat, Rajdip Dhillon, Jane Edwards, Javier Macias-Guarasa, Nelson Morgan, Barbara Peskin, Elizabeth Shriberg, Andreas Stolcke, Chuck Wooters, and Britta Wrede. 2004. The ICSI Meeting Project: Resources and Research. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), Meeting Recognition Workshop*, Montreal, Quebec, Canada.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*, pages 25–32, Sydney, Australia.

Jane Morris and Graeme Hirst. 1991. Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. *Computational Linguistics*, 17(1):21–48.

Matthew Purver, Konrad P. Körding, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2006. Unsupervised Topic Modelling for Multi-Party Spoken Discourse. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL)*, pages 17–24, Sydney, Australia.

Jeffrey Reynar. 1994. An Automatic Method of Finding Topic Boundaries. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 331–333, Las Cruces, New Mexico, USA.

Exploiting Unlabeled Text to Extract New Words of Different Semantic Transparency for Chinese Word Segmentation

Richard Tzong-Han Tsai[†*] and **Hsi-Chuan Hung**[‡]

[†]Department of Computer Science and Engineering,
Yuan Ze University, Chung-Li, Taoyuan, Taiwan
[‡]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
thtsai@saturn.yzu.edu.tw yabthung@gmail.com
[*]corresponding author

Abstract

This paper exploits unlabeled text data to improve new word identification and Chinese word segmentation performance. Our contributions are twofold. First, for new words that lack semantic transparency, such as person, location, or transliteration names, we calculate association metrics of adjacent character segments on unlabeled data and encode this information as features. Second, we construct an internal dictionary by using an initial model to extract words from both the unlabeled training and test set to maintain balanced coverage on the training and test set. In comparison to the baseline model which only uses n-gram features, our approach increases new word recall up to 6.0%. Additionally, our approaches reduce segmentation errors up to 32.3%. Our system achieves state-of-the-art performance for both the closed and open tasks of the 2006 SIGHAN bakeoff.

1 Introduction

Many Asian languages do not delimit words by spaces. Word segmentation is therefore a key step for language processing tasks in these languages. Chinese word segmentation (CWS) systems can be built by supervised learning from a labeled data set. However, labeled data sets are expensive to prepare as it involves manual annotation efforts. Therefore, exploiting unlabeled data to improve CWS performance becomes an important research goal. In addition, new word identification (NWI) is also very important because they represent the latest information, such as new product names.

This paper explores methods of extracting information from both internal and external unlabeled data to augment NWI and CWS. According to (Tseng and Chen, 2002), new words can be divided into two major categories: Words with high or low semantic transparency (ST), which describes the correlation of semantic meanings between a word and its morphemes. We designed effective strategies toward the identification of these two new word types. One is based on transductive learning and the other is based on association metrics.

2 The Model

2.1 Formulation

We convert the manually segmented words into tagged character sequences. We tag each character with either B, if it begins a word, or I, if it is inside or at the end of a word.

2.2 Conditional Random Fields

CRFs are undirected graphical models trained to maximize a conditional probability (Lafferty et al., 2001). A linear-chain CRF with parameters $\Lambda = \lambda_1, \lambda_2, \ldots$ defines a conditional probability for a state sequence $y = y_1 \ldots y_T$ given an input sequence $x = x_1 \ldots x_T$ to be

$$P_\Lambda(y|x) = \frac{1}{f_x} \exp\left(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(y_{t-1}, y_t, x, t)\right)$$

where Z_x is the normalization that makes the probability of all state sequences sum to one; $f_k(y_{t-1}, y_t, x, t)$ is often a binary-valued feature function and λ_k is its weight. The feature functions can measure any aspect of a state transition, $y_{t-1} \rightarrow y_t$, and the entire observation sequence, x, centered at the current position, t. For example, one feature function might have value 1 when y_{t-1} is the state B, y_t is the state I, and is the character "國". Large positive values for λ_k indicate a preference for such an event; large negative values make the event unlikely.

In our CRF model, each binary feature is multiplied with all states (y_t) or all state transitions ($y_{t-1}y_t$). For simplicity, we omit them in the following discussion. In addition, we use C_0 rather than x_t to denote the current character.

3 Baseline n-gram Features

Character n-gram features have proven their effectiveness in ML-based CWS (Xue and Shen, 2003). We use 4 types of unigram feature functions: C_0, C_1 (next character), C_{-1} (previous character), C_{-2} (character preceding C_{-1}). Furthermore, 6 types of bigram features are used, and are designated here as conjunctions of the previously specified unigram features, $C_{-2}C_{-1}$, $C_{-1}C_0$, C_0C_1, $C_{-3}C_{-1}$, $C_{-2}C_0$, and $C_{-1}C_1$.

4 New Word Identification

We mainly focus on improving new word identification (NWI) using unlabeled text. Words with high and low ST are discussed separately due to the disparity in their morphological characteristics. However, it is unnecessary for our system to classify words as high- or low-ST because our strategies for dealing with these two classes are employed synchronously.

4.1 High-ST words

For a high-ST word, its meaning can be easily derived from those of its morphemes. A word's semantic meaning correlates to its tendency of being affixed to longer words. This behavior can be recorded by the baseline n-gram model. When the baseline model is used to segment a sentence containing a high-ST word, since this tendency is consistent with that is recorded in the baseline model, this word tends to be successfully segmented. For example, suppose 指南車 zhi-nan-che (compass chariot) is in the training set. The baseline n-gram model will record the tendency of 指南 zhi-nan (guide) that it tends to be a prefix of a longer word. When tagging a sentence that contains another high ST word also containing 指南, such as 指南針 zhi-nan-zhen (compass), this word can be correctly identified.

Using only n-gram features may prevent some occurrences of high-ST words from being identified due to the ambiguity of neighboring n-grams. To rectify this problem, we introduce the transductive dictionary (TD) feature, which is similar to the traditional dictionary feature that indicates if a sequence of characters in a sentence matches a word w in an existing dictionary. The difference is that the TD not only comprises words in the training set, but contains words extracted from the unlabeled test set. The transductive dictionary is so named because it is generated following general concepts of transductive learning. We believe adding TD features can boost recall of high-ST words. More details on the TD are found in Section 5. The TD features that identify high-ST words are detailed in Section 6.1.

4.2 Low-ST words

On the contrary, new words lack of ST, such as transliteration names, are more likely to be missed by the baseline n-gram model, because their morphemes' morphological tendencies are not guaranteed to be consistent with those recorded by n-gram features. For instance, suppose 天平 tian-ping (libra) only appears as individual words in the training set. The baseline model cannot identify 熊天平 xiong-tian-ping (a singer's name) because 熊天平 is a low-ST word and the morphological tendency of 天平 is not consistent with the recorded one.

In English, there is a similar phenomenon called multi-word expressions (MWEs). (Choueka, 1988) regarded MWE as connected collocations: a sequence of neighboring words "whose exact meaning cannot be derived from the meaning or connotation of its components", which means that MWEs also have low ST. As some pioneers provide MWE identification methods which are based on association metrics (AM), such as likelihood ratio (Dunning, 1993).

The methods of identifying low-ST words can be divided into two: filtering and merging. The former uses AM to measure the likelihood that a candidate is actually a whole word that cannot be divided. Candidates with AMs lower than the threshold are filtered out. The latter strategy merges character segments in a bottom-up fashion. AMs are employed to suggest the next candidates for merging. Both methods suffer from two main drawbacks of AM: dependency on segment length and inability to use relational information between context and tags. In the first case, applying AMs to ranking character segment pairs, it is difficult to normalize the values calculated from pairs of character segments of various lengths. Secondly, AMs ignore the relationships among n-grams (or other contextual information) and labels, which are abundant in annotated corpora, and they only use annotation data to determine thresholds. In Section 6.2, we illustrate how encoding AMs as features can avoid the above weaknesses.

5 Balanced Transductive Dictionary

The simplest TD is composed of words in the training set and words extracted from the unlabeled test set. The main problem of such TD is the disparity in training and test set coverage. During training, since its coverage is 100%, the enabled dictionary features will be assigned very high weights while n-gram features

will be assigned low weights. During testing, when coverage is approximately 80-90%, most tags are decided by dictionary features enabled by IV words, while n-gram features have little influence. As a result, it is likely that only IV words are correctly segmented, while OOV words are over-segmented. Loosely speaking, a dictionary's coverage of the training set is linked to the degree of reliance placed by the CRF model on the corresponding dictionary features. Therefore, the dictionary should be made more balanced in order to avoid the potential problem of overfitting. Here a dictionary is said to be more balanced if its coverage of the training set approximates its coverage of the test set while maximizing the latter. Afterward we name the TD composed of words from gold training set and tagged test set and as Naïve TD (NTD) for its unbalanced coverage in training and test set.

Our TD is constructed as follows. Given *initial features*, we use the model trained on the whole training set with these features to label the test set and add all words into our TD.

The next step is to balance our TD's coverage of the training and test sets. Since coverage of the test set cannot reach 100%, the only way to achieve this goal is by slightly lowering the dictionary's coverage on the training set. We apply n-fold cross-tagging to label the training set data: Each fold that has $1/n$ of the training set is tagged by the model trained on the other $n-1$ folds with initial features. All the words identified by this cross-tagging process are then added to our TD. The difference between the NTD and our TD is that the NTD extracts words from the gold training set, but our TD extracts words from the cross-tagged training set. Finally, our TD is used to generate dictionary features to train the final model. Since the TD constructed from cross-tagging training set and tagged test set exists more balanced coverage of the training and test set, we call such a TD "balanced TD", shorted as BTD.

6 Our NWI Features

6.1 Transductive Dictionary Features

If a sequence of characters in a sentence matches a word w in an existing dictionary, it may indicate that the sequence of characters should be segmented as one word. The traditional way is to encode this information as binary word match features. To distinguish the matches with the same position and length, we propose a new word match feature that contains frequency information to replace the original binary word match feature. Since over 90% of words are four or fewer characters in length, we only consider words of one to four characters. In the following sections, we use D to denote the dictionary.

6.1.1 Word Match Features (WM)

This feature indicates if there is a sequence of neighboring characters around C_0 that match a word in D. Features of this type are identified by their positions relative to C_0 and their lengths. Word match features are defined as:

$$\mathrm{WM}(w = C_{-pos} \ldots C_{-pos+len-1}) = \begin{cases} 1 & \text{if } w \in D \\ 0 & \text{otherwise} \end{cases}$$

where $len \in [1..4]$ is w's length and $pos \in [0..len]$ is C_0's zero-based relative position in w (when $pos = len$, the previous len characters form a word found in D). If C_0 is "會" and "討論會" is found in D, $\mathrm{WM}(C_{-2} \ldots C_0)$ is enabled.

6.1.2 Word Match with Word Frequency (WMWF)

Given two different words that have the same position and length, WM features cannot differentiate which should have the greater weight. This could cause problems when two matched words of same length overlap. (Chen and Bai, 1998) solved this conflict by selecting the word with higher (frequency × length). We utilize this idea to reform the WM features into our WMWF features:

$$\mathrm{WMWF}_q(w = C_{-pos} \ldots C_{-pos+len-1}) = \begin{cases} 1 & \text{if } w \in D \text{ and } \mathrm{log_feq}(w) = q \\ 0 & \text{otherwise} \end{cases}$$

where the word frequency is discretized into 10 bins in a logarithmic scale:$\mathrm{log_feq}(w) = \min(\lceil \log_2 w\text{'s frequency} + 1 \rceil, 10)$

thus $q[0..10]$ is the discretized log frequency of w. In this formulation, matching words with higher log frequencies are more likely to be the correct segmentation. Following the above example, if the frequency of "討論會" is 15, then the feature $\mathrm{WMWF}_4(C_{-2} \ldots C_0)$ is enabled.

6.1.3 Discretization v.s. Zipf's Law

Since current implementations of CRF models only allow discrete features, the word frequency must be discretized. There are two commonly used discretization methods: equal-width interval and equal-frequency interval, where the latter is shown to be more suitable for data following highly skewed distribution (Ismail and Ciesielski, 2003). The word frequency distribution is the case: Zipf's law (Zipf, 1949) states that the word frequency is inversely proportional to its rank (Adamic and Huberman, 2002):

$$f(x) \propto z^{-\alpha}$$

where $f(x)$ is x's frequency, z is its rank in the frequency table, and α is empirically found to be close to unity. Obviously this distribution is far from flat uniform. Hence the equal-frequency binning turns out to be our choice.

Ideally, we would like each bin to have equal *expected* number of values rather than following *empirical* distribution. Therefore, we attempt to discretize according to their underlying Zipfian distribution.

Adamic & Huberman (2002) shows that Zipf's law is equivalent to the power law, which describes Zipf's law in a unranked form:

$$f_X(x) \propto x^{-(1+(1/\alpha))},$$

where X is the random variable denoting the word frequency and $f_X(x)$ is its probability density function. Approximated by integration, the expected number of values in the bin $[a, b]$ can be calculated as

$$\sum_{a\le x\le b} x \cdot \Pr[X = x] \approx \int_a^b x \cdot f_X(d)dx$$
$$\propto \int_a^b x \cdot x^{-(1+(1/\alpha))}dx \approx \ln x|_a^b = \ln(b/a)$$
$$(\because \alpha \approx 1)$$

Thus each bin has equal number of values within it if and only if b/a is a constant, which is in a log scale. This shows that our strategy to discretize the WMWF and WMNF features in a log scale is not only a conventional heuristic but also has theoretical support.

6.2 Association Metric Features (AM)

In this section, we describe how to formulate the association metrics as features to avoid the weakness stated in Section 4.2. Our idea is to enumerate all possible character segment pairs before and after the segmentation point and treat their association metrics as feature values. Each possible pair corresponds to an individual feature. For computational feasibility, only pairs with total length shorter than five characters are selected. All the enumerated segment pairs are listed in the following table:

Feature	x,y	Feature	x,y
AM^{1+1}	c_{-1}, c_0	AM^{2+1}	$c_{-2}c_{-1}$, c_0
AM^{1+2}	c_{-1}, c_0c_1	AM^{2+2}	$c_{-2}c_{-1}$, c_0c_1
AM^{1+3}	c_{-1}, $c_0c_1c_2$	AM^{3+1}	$c_{-3}c_{-2}c_{-1}$, c_0

We use Dunning's method (Dunning, 1993) because it does not depend on the assumption of normality and it allows comparisons to be made between the significance of the occurrences of both rare and common phenomenon. The likelihood ratio test is applied as follows:

$$\begin{aligned} LR(x,y) =& 2 \times (\mathrm{logl}(p_1,k_1,n_1) + \mathrm{logl}(p_2,k_2,n_2) \\ & - \mathrm{logl}(p,k_1,n_1) - \mathrm{logl}(p,k_2,n_2)) \end{aligned}$$

where $\mathrm{logl}(P,K,M) = K \times \ln P + (M-K) \times \ln(1-P)$, $k_1 = \mathrm{freq}(x,y)$; $k_2 = f(x,\neg y) = \mathrm{freq}(x)-k_1$; $n_1 = \mathrm{freq}(y)$; $n_2 = N - n_1$; $p_1 = \mathrm{p}(x|y) = k_1/n_1$; $p_2 = \mathrm{p}(x|y) = k_2/n_2$; $p = \mathrm{p}(x) = (k_1+k_2)/N$;
N is the number of words in corpus.

An important property of likelihood ratio is that $-2LR$ is asymptotically x_1^2 distributed. Hence we can directly compute its p-value. We then discretize the p-value into several bins, each bin is defined by two significance levels $2^{-(q+1)}$ and 2^{-q}. Thus, our AM feature is defined as:

$$AM_q(x,y) = \begin{cases} 1 & \text{if the } p\text{-value of} \\ & LR(x,y) \in [2^{-(q+1)}, 2^{-q}] \\ 0 & \text{otherwise} \end{cases}$$

Since we have a constraint $0 \le q \le 10$, thus, the last interval is $[0, 2^{-10}]$. We can think that larger q implies higher tendency of current character to be labeled as 'I'.

7 External Dictionary Features

7.1 Word Match with Ngram Frequency (WMNF)

In addition to internal dictionaries extracted from the training and test data, external dictionaries can also be used. Unlike with internal dictionaries, the true frequency of words in external dictionaries cannot be acquired. We must treat each external dictionary word as an n-gram and calculate its frequency in the entire unsegmented (training plus test) set as follows:

$$\begin{aligned} &\mathrm{WMNF}_q(w = C_{-pos}\ldots C_{-pos+len-1}) \\ &= \begin{cases} 1 & \text{if } w \in D \text{ log_ngram_freq}(w) = q \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where the frequencies are discretized into 10 bins by the same way describing in previous section.

In this formulation, matching n-grams with higher log frequencies are more likely to represent correct segmentations.

8 Experiments and Results

8.1 Data and Evaluation Metrics

We use two datasets in SIGHAN Bakeoff 2006: one Simplified Chinese provided by Univ. of Pennsylvania (UPUC) and one Traditional Chinese provided by the City Univ. of HK (CITYU), as shown in Table 1.

Two unlabeled text data used in our experiments. For the CITYU dataset, we use part of the CIRB40 corpus[1] (134M). For the UPUC dataset, we use the Contemporary Chinese Corpus at PKU[2] (73M).

[1] http://clqa.jpn.org/2006/04/corpus.html
[2] http://icl.pku.edu.cn/icl_res/

			UPUC							CITYU						
			F	+/-	R_{oov}	+/-	R_{iv}	NC	NCRR	F	+/-	R_{oov}	+/-	R_{iv}	NC	NCRR
closed	1	N-grams	93.0	n/a	71.1	n/a	95.7	14094	n/a	96.6	n/a	78.8	n/a	97.3	9642	n/a
	2	(1) + AM (int_raw)	94.3	+1.3	76.4	+5.3	96.5	11655	+17.3	97.3	+0.7	80.3	+1.5	97.9	7890	+18.2
	3	(1) + WM, NTD(1)	93.4	+0.4	74.8	+3.7	95.4	13182	+6.5	97.0	+0.4	81.6	+2.8	97.3	8597	+10.8
	4	(1) + WMWF, NTD(1)	93.7	+0.7	75.0	+3.9	95.8	12719	+9.7	97.2	+0.6	82.0	+3.2	97.6	8029	+16.7
	5	(1) + WMWF, BTD(1)	94.0	+1.0	73.4	+2.3	96.7	12218	+13.3	97.4	+0.8	79.2	+0.4	98.3	7429	+23.0
	6	(1) + WMWF, BTD(2) + AM (int_raw)	94.5	+1.5	76.6	+5.5	96.7	11173	+20.7	97.5	+0.9	80.3	+1.5	98.2	7377	+23.5
	7	Rank 1 in Closed	93.3	n/a	70.7	n/a	96.3	n/a	n/a	97.2	n/a	78.7	n/a	98.1	n/a	n/a
open	8	(1) + AM (ext_raw)	94.3	+1.3	75.9	+4.8	96.6	11695	+17.0	97.3	+0.7	81.9	+3.1	97.9	7747	+19.7
	9	(1) + WMWF, BTD(8) + AM (ext_raw)	94.7	+1.7	77.1	+6.0	96.9	10844	+23.1	97.8	+1.2	82.2	+3.4	98.5	6531	+32.3
	10	(9) + WMNF	95.0	+2.0	78.7	+7.6	97.1	10326	+26.7	97.9	+1.3	84.0	+5.2	98.5	6117	+36.6
	11	Rank 1 in Open	94.4	n/a	76.8	n/a	96.6	n/a	n/a	97.7	n/a	84.0	n/a	98.4	n/a	n/a

Table 2: Comparison scores for UPUC and CITYU

Source	Training (Wds/Types)	Test (Wds/Types)
UPUC	509K/37K	155K/17K
CITYU	1.6M/76K	220K/23K

Table 1: An overview of corpus statistics

We use SIGHAN's evaluation script to score all segmentation results. This script provides three basic metrics: Precision (P), Recall (R), and F-Measure (F). In addition, it also provides three detailed metrics: R_{OOV} stands for the recall rate of the OOV words. R_{IV} stands for the recall rate of the IV words, and NC stands for NChanges (insertion+deletion+substitution) (Sproat and Emerson, 2003). In addition, we also compare the NChange reduction rate (NCRR) because the CWS's state-of-the art F-measure is over 90%. Here, the NCRR of any system s is calculated:

$$\text{NCRR}(s) = \frac{NChange_{baseline} - NChange_s}{NChange_{baseline}}$$

8.2 Results

Our system uses the n-gram features described in Section 3 as our baseline features, denoted as n-grams. We then sequentially add other features and show the results in Table 2. Each configuration is labeled with the features and the resources used in it. For instance, AM(int_raw) means AM features computed from the internal raw data, including the unlabeled training and test set, and WM, NTD(1) stands for WM features based on the NTD employing config.1's feature as its initial features.

Our experiments are conducted in the following order: starting from baseline model, we then gradually add AM features (config.2) and TD features (config.4 & 5) and combined them as our final setting (config.6) for the closed task. In the open task, we sequentially add AM features (config.8), TD features (config.9), which only exploit internal and unlabeled data. Finally, the last setting (config.10) employs external dictionaries besides all above features.

Association Metric At first, we compare the effects after adding AM which is computed based on the internal raw data (config.2). We can see that adding AM can significantly improve the performance on both datasets. Also, the OOV-recall is improved 5.3% and 1.5% on UPUC and CITYU respectively.

Transductive Dictionary Without lost of generality, we firstly use the WM features introduced in Section 6.1.1 to represent dictionary features which is denoted as config. 3 in Tables 3. We can see that the configuration with WM features outperforms that with N-grams (config.1). It is worth mentioning that even though N-grams achieve satisfactory OOV recall (0.788 and 0.711) in CITYU and UPUC, config. 3 achieves higher OOV recall.

Frequency Information and BTD To show the effectiveness of frequency, we compare WM with WMWF features. In Table 2, we can see that WMWF features (config.4) outperform WM features (config.3) on both datasets in terms of F-Measure and R_{IV}. In addition, switching the NTD (config.4) with BTD (config.5) can further improve R_{IV} and F-score while R_{OOV} slightly decreases. This is not surprising. In a BTD, most incorrectly segmented words appear infrequently. Unfortunately, the new words detected by the baseline model also have comparatively low frequencies. Therefore, these words will be assigned into the same several bins corresponding to infrequent words as the incorrectly segmented words and share low weights with them.

Combined Effects In config.6, we use the model with N-gram plus AM features as initial features to construct the BTD. In Table 2, we can see that the increase of R_{OOV}'s can recover the loss brought by using BTD and further raise

the F-measure to the level of the state-of-the-art open task performance.

In comparison of the baseline n-gram model, our approach reduces the errors by an significant number of 20.7% and 23.5% in the UPUC and CITYU datasets, respectively. The OOV recall of our approach increases 5.5% and 1.5% on the UPUC and CITYU datasets, respectively. Astonishingly, in the UPUC dataset, with limited information provided by training corpus and unlabeled test data, our system still outperforms the best SIGHAN open CWS system that are allowed to use unlimited external resources.

8.2.1 Using External Unlabeled Data

In config.9, we also use the ngrams plus AM as initial features to generate the BTD, but external unlabeled data are used along with internal data to calculate values of AM features. Comparing with config.6, we can see that R_{OOV}, R_{IV}, and F-score are further improved, especially R_{OOV}. Notably, this configuration can reduce NChanges by 2.4% in comparison of the best closed configuration.

8.2.2 Using External Dictionaries

To demonstrate that our approach can be expandable by installing external dictionaries, we add WMNF features based on the external dictionaries into the config.9, and denote this to be our config.10. We use the Grammatical Knowledge-Base of Contemporary Chinese (GKBCC) (Yu et al., 2003) and Chinese Electronic Dictionary for the UPUC and CITYU dataset, respectively.

As shown in Table 2, all metrics of config.10 are better than config.9, especially R_{OOV}. This is because most of the new words do not exist in external dictionaries; therefore, using external dictionaries can complement our results.

9 Conclusion

This paper presents how to exploit unlabeled data to improve both NWI and CWS performance. For new high-ST words, since they can be decomposed into semantically relevant atomic parts, they could be identified by the n-gram models. Using the property, we construct an internal dictionary by using this model to extract words from both the unlabeled training and test set to maintain balanced coverage on them, which makes the weights of the internal dictionary features more accurate. Also, frequency is initiatively considered in dictionary features and shows its effectiveness.

For low-ST words, we employ AMs, which is frequently used in English MWE extraction to enhance the baseline n-gram model. We show that this idea effectively extract much more unknown person, location, and transliteration names which are not found by the baseline model.

The experiment results demonstrate that adopting our two strategies generally beneficial to NWI and CWS on both traditional and simplified Chinese datasets. Our system achieves state-of-the-art closed task performance on SIGHAN bakeoff 2006 datasets. Under such most stringent constraints defined in the closed task, our performances are even comparable to open task performance. Moreover, with only external unlabeled data, our system also achieves state-of-the-art open task performance on SIGHAN bakeoff 2006 datasets.

References

L.A. Adamic and B.A. Huberman. 2002. Zipf's law and the internet. *Glottometrics*, 3:143–150.

K. J. Chen and M. H. Bai. 1998. Unknown word detection for chinese by a corpus-based learning method. *Computational Linguistics and Chinese Language Processing*, 3(1):27–44.

Y. Choueka. 1988. Looking for needles in a haystack or locating interesting collocation expressions in large textual databases. In *RIAO*.

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):65–74.

Michael K. Ismail and Vic Ciesielski. 2003. An empirical investigation of the impact of discretization on common data distributions. In *Design and Application of Hybrid Intelligent Systems*. IOS Press, Amsterdam, Netherlands.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-01*, pages 282–289.

Richard Sproat and Thomas Emerson. 2003. The first international chinese word segmentation bakeoff. In *SIGHAN-03*.

Huihsin Tseng and Keh-Jiann Chen. 2002. Design of chinese morphological analyzer. In *SIGHAN-02*.

Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *SIGHAN-03*.

G.K. Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA.

Social Network Inspired Models of NLP and Language Evolution

Monojit Choudhury
Microsoft Research Lab India
196/36 2nd Main, Sadashivnagar, Bangalore, India 560080
monojitc@microsoft.com

Animesh Mukherjee and **Niloy Ganguly**
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India 721302
{animesh,niloy}@cse.iitkgp.ernet.in

Abstract

Human language with all its intricacies is perhaps one of the finest examples of a complex system. Therefore, it becomes absolutely necessary to study the faculty of language from the perspective of a complex system. Of late, there has been an upsurge in the use of networks in modeling the complex dynamics of various natural and artificial systems. While some of these works aim at using social network techniques to build certain end-user applications, others are more fundamental in the sense that they employ these techniques to explain the emergent properties of a complex system as a whole. A substantial amount of research have also been done in the field of linguistics to employ social networks in the design of efficient solutions for numerous problems in NLP and language evolution. The objective of this tutorial is to show how language and its dynamics can be successfully studied in the framework of social networks. The tutorial will particularly demonstrate the relevance of social network-based methods in the development of a large variety of NLP applications and in understanding the dynamics of language evolution and change.

The tutorial is divided into two parts. Part I begins with a brief introduction to this field showing how linguistic entities and the interactions between them can be respectively represented through the nodes and edges of a network. This will be followed by a comprehensive survey of the general theory of social networks with a special emphasis on the methods of analysis and models of synthesis for such networks.

Part II presents three case studies. The first case study is on unsupervised POS tagging, the second one involves modeling of the mental lexicon and applications of such models in spell checking and word sense disambiguation. The third case study demonstrates the usefulness of social networks in explaining some of the evolutionary dynamics of language pertaining to the sound inventories.

The tutorial is concluded by (a) comparing the above methods with more traditional methods of doing NLP, (b) providing pointers as to where to look for/publish materials in this area, and, (c) indicating some of the future research directions.

Biography

Monojit Choudhury is a post doctoral researcher at Microsoft Research, India. He has submitted his PhD from the Department of Computer Science and Engineering, IIT Kharagpur and earlier, received his B.Tech from the same department. Mr. Choudhury received the *Young Scientist Award* from the Indian Science Congress Association in 2003.

Animesh Mukherjee is a PhD student in the Department of Computer Science and Engineering, IIT Kharagpur and also a Microsoft Research fellow. He received his MTech from the same department, and BTech from Haldia Institute of Technology. Mr. Mukherjee received the *Young Scientist Award* from the Indian Science Congress Association in 2006.

Niloy Ganguly is an assistant professor in the Department of Computer Science and Engineering, IIT Kharagpur. He has received his PhD in Computer Science from Bengal Engineering and Science University, Calcutta and his Bachelors from IIT Kharagpur. He was a post doctoral fellow in Technical University of Dresden, Germany. He has numerous publications in international journals and conferences including ACL, PODC, SIGCOMM, ACM and IEEE Trans.

How to Add a New Language on the NLP Map: Building Resources and Tools for Languages with Scarce Resources

Rada Mihalcea
University of North Texas
rada@cs.unt.edu

Vivi Nastase
EML Research gGmbH
Vivi.Nastase@eml-r.villa-bosch.de

Abstract

Those of us whose mother tongue is not English or are curious about applications involving other languages, often find ourselves in the situation where the tools we require are not available. According to recent studies there are about 7200 different languages spoken worldwide – without including variations or dialects – out of which very few have automatic language processing tools and machine readable resources.

In this tutorial we will show how we can take advantage of lessons learned from frequently studied and used languages in NLP, and of the wealth of information and collaborative efforts mediated by the World Wide Web. We structure the presentation around two major themes: mono-lingual and cross-lingual approaches. Within the mono-lingual area, we show how to quickly assemble a corpus for statistical processing, how to obtain a semantic network using on-line resources – in particular Wikipedia – and how to obtain automatically annotated corpora for a variety of applications. The cross-lingual half of the tutorial shows how to build upon NLP methods and resources for other languages, and adapt them for a new language. We will review automatic construction of parallel corpora, projecting annotations from one side of the parallel corpus to the other, building language models, and finally we will look at how all these can come together in higher-end applications such as machine translation and cross-language information retrieval.

Biographies

Rada Mihalcea is an Assistant Professor of Computer Science at the University of North Texas. Her research interests are in lexical semantics, multilingual natural language processing, minimally supervised natural language learning, and graph-based algorithms for natural language processing. She serves on the editorial board of the Journal of Computational Linguistics, the Journal of Language Resources and Evaluations, the Journal of Natural Language Engineering, the Journal of Research in Language in Computation, and the recently established Journal of Interesting Negative Results in Natural Language Processing and Machine Learning.

Vivi Nastase is a post-doctoral fellow at EML Research gGmbH, Heidelberg, Germany. Her research interests are in lexical semantics, semantic relations, knowledge extraction, multi-document summarization, graph-based algorithms for natural language processing, multilingual natural language processing. She is a co-founder of the Journal of Interesting Negative Results in Natural Language Processing and Machine Learning.

Introduction to Text Summarization and Other Information Access Technologies

Horacio Saggion
Natural Language Processing Group
University of Sheffield
211 Portobello Street - Sheffield - S1 4DP
England - United Kingdom
saggion@dcs.shef.ac.uk

Abstract

In recent years we have witnessed an explosion of on-line unstructured information in multiple languages, making natural language processing technologies such as automatic text summarization increasingly important for the information society. Text Summarization provides users with condensed descriptions of documents, allowing them to make informed decisions based on text summaries. Text summarization can be combined with Information Retrieval (IR) and Question Answering (QA) to provide users with focus-based or query-based summaries which are targeted towards the users' specific needs. When the information a user looks for is spread across multiple sources, text summarization can be used to condense facts and present a non-redundant account of the most relevant facts found across a set of documents.

The objective of this IJCNLP 2008 tutorial is to give an overview of a number of technologies in natural language processing for information access including: single and multi-document summarization, cross-lingual summarization; and summarization in the context of question answering.

The tutorial will discuss summarization concepts and techniques as well as its relation and relevance to other technologies such as information retrieval and question answering. It will also include description of available resources for development, training and evaluation of summarization components. A summarization (multi-document and multilingual) toolkit will be used for demonstration purposes. A number of question answering components relevant for the creation of definitional summaries and profiles will also be demonstrated.

Biography

Dr. Saggion is a research fellow in the Natural Language Processing group, Department of Computer Science, University of Sheffield, England, UK. His area of expertise is Text Summarization. He works in national and international projects on information extraction, ontology-based information extraction, question answering, and text summarization. He obtained his PhD. in 2000 from Université de Montréal, Departement d'Informatique et de Recherche Operationnelle. He has published over 50 works in conferences, workshops and journal papers. Together with his research career, he has been an active teacher, he was assistant professor and researcher at Universidad de Buenos Aires, and Universidad Nacional de Quilmes, and teaching assistant at Université de Montréal. He has participated in a number of summarization and question answering evaluations including DUC 2004, DUC 2005, MSE 2005, TREC/QA 2003, TREC/QA 2004, TREC/QA 2005. He has recently organised the workshops "Multi-source Multi-lingual Information Extraction and Summarization" and "Crossing Barriers in Text Summarization Research" in the RANLP Conferences. He has given courses and tutorials on Text Summarization and other technologies such as question answering in a number of international venues such as ESSLLI and LREC.

A Punjabi Grammar Checker

Mandeep Singh Gill
Department of Computer Science
Punjabi University
Patiala -147002, India
msgill_in@yahoo.com
Tel.: +91-9888165971

Gurpreet Singh Lehal
Department of Computer Science
Punjabi University
Patiala -147002, India
gslehal@yahoo.com
Tel.: +91-175-3046171

Shiv Sharma Joshi
Department of Anthropological Linguistics & Punjabi Lexicography
Punjabi University
Patiala -147002, India
Tel.: +91-175-3046292

Abstract

This article provides description about the grammar checking software developed for detecting the grammatical errors in Punjabi texts and providing suggestions wherever appropriate to rectify those errors. This system utilizes a full-form lexicon for morphology analysis and rule-based systems for part of speech tagging and phrase chunking. The system supported by a set of carefully devised error detection rules can detect and suggest rectifications for a number of grammatical errors, resulting from lack of agreement, order of words in various phrases etc., in literary style Punjabi texts.

1 Introduction

Grammar checking is one of the most widely used tools within natural language engineering applications. Most of the word processing systems available in the market incorporate spelling, grammar, and style-checking systems for English and other foreign languages, one such rule-based grammar checking system for English is discussed in (Naber, 2003). However, when it comes to the smaller languages, specifically the Indian languages, most of such advanced tools have been lacking. Spell checking has been addressed for most of the Indian languages but still grammar and style checking systems are lacking. In this article a grammar checking system for Punjabi, a member of the Modern Indo-Aryan family of languages, is provided. The grammar checker uses a rule-based system to detect grammatical errors in the text and if possible generates suggestions to correct those errors.

To the best of our knowledge the grammar checking provided here will be the first such system for Indian languages. There is n-gram based grammar checking system for Bangla (Alam et al, 2006). The authors admit its accuracy is very low and there is no description about whether the system provides any suggestions to correct errors or not. It is mentioned that it was tested to identify correct sentences from the set of sentences provided as input but nothing is mentioned as far as correcting those errors is concerned. However, the system that we discuss here for Punjabi detects errors and suggests corrections as well. In doing so, provides enough information for the user to understand the error reason and supports the suggestions provided, if any.

2 System Overview

The input Punjabi text is given to the preprocessing system that performs tokenization and detects any phrases etc. After that morphological analysis is performed, this returns possible tags for all the words in the given text, based on the full-form lexicon that it is using. Then a rule-based part of speech tagger is engaged to disambiguate the tags based on the context information. After that, the text is grouped into various phrases accordingly to the pre-defined phrase chunking rules. In the final phase, rules to check for various grammatical errors internal to phrases and agreement on the sentence level, are applied. If any error is found in a sentence then based on the context information corrections are suggested (generated) for that.

For the purpose of morphological analysis we have divided the Punjabi words into 22 word classes like noun, adjective (inflected and uninflected), pronoun (personal, demonstrative, reflexive, interrogative, relative, and indefinite), verb (main verb, operator verb, and auxiliary verb), cardinals, ordinals, adverb, postposition, conjunction, interjection etc., depending on the grammatical information required for the words of these word classes. The information that is in the database depends upon the word class, like for noun and inflected adjective, it is gender, number, and case. For personal pronouns, person is also required. For main verbs gender, number, person, tense, phase, transitivity etc. is required. As mentioned earlier the lexicon of this morphological analyzer is full form based i.e. all the word forms of all the commonly used Punjabi words are kept in the lexicon along with their root and other grammatical information.

For part of speech tagging, we have devised a tag set keeping into mind all the grammatical categories that can be helpful for agreement checking. At present, there are more than 600 tags in the tag set. In addition to this, some word-specific tags are also there. The tag set is very user friendly and while choosing tag names existing tag sets for English and other such languages were taken into consideration, like NNMSD – masculine, singular, and direct case noun, PNPMPOF – masculine, plural, oblique case, and first person personal pronoun. The approach followed for part of speech tagging is rule-based, as there is no tagged corpus for Punjabi available at present. As the text we are processing may have grammatical agreement errors, so the part of speech tagging rules are devised considering this. The rules are applied in sequential order with each rule having an attached priority to control its order in this sequence.

For phrase chunking, again a rule-based approach was selected mainly due to the similar reasons as for part of speech tagging. The tag set that is being used for phrase chunking includes tags like NPD – noun phrase in direct case, NPNE – noun phrase followed by ਨੇ ne etc. The rules for phrase chunking also take into account the potential errors in the text, like lack of agreement in words of a potential phrase. However, as would be expected there is no way to take the misplaced words of a phrase into account, like if words of a phrase are separated (having some other phrase in between) then that cannot be taken as a single phrase, even though this may be a potential error.

In the last phase i.e. grammar checking, there are again manually designed error detection rules to detect potential errors in the text and provide corrections to resolve those errors. For example, rule to check modifier and noun agreement, will go through all the noun phrases in a sentence to check if the modifiers of those sentences agree with their respective head words (noun/pronoun) in terms of gender, number, and case or not. For this matching, the grammatical information from the tags of those words is used. In simple terms, it will compare the grammatical information (gender, number, and case) of modifier with the headword (noun/pronoun) and displays an error message if some grammatical information fails to match. To resolve this error, the grammar checking module will use morphological generator, to generate the correct form (based on headword's gender, number, and case) for that modifier from its root word.

For example, consider the grammatically incorrect sentence ਸੋਹਣੇ ਲੜਕਾ ਜਾਂਦਾ ਹੈ sohne larka janda hai 'handsome boy goes'. In this sentence in the noun phrase, ਸੋਹਣੇ ਲੜਕਾ sohne larka 'handsome boy', the modifier ਸੋਹਣੇ sohne 'handsome' (root word – ਸੋਹਣਾ sohna 'handsome'), with masculine gender, plural number, and direct case, is not in accordance with the gender, number, case of its head word. It should be in singular number instead of plural. The grammar checking module will detect this as an error as 'number' for modifier and headword is not same, then it will use morphological generator to generate the 'singular number form' from its root word, which is same as root form i.e. ਸੋਹਣਾ sohna 'handsome' (masculine gender, singular number, and direct case). So, the input sentence will be corrected as ਸੋਹਣਾ ਲੜਕਾ ਜਾਂਦਾ ਹੈ sohna larka janda hai 'handsome boy goes'.

The error detection rules in grammar checking module are again applied in sequential order with priority field to control the sequence. This is done to resolve phrase level errors before going on to the clause level errors, and then to sentence level agreement errors.

3 Grammar Errors

At present, this grammar checking system for Punjabi detects and provides corrections for following grammatical errors, based on the study of Punjabi grammar related texts (Chander, 1964; Gill and Gleason, 1986; Puar, 1990):

Modifier and noun agreement
The modifier of a noun must agree with the noun in terms of gender, number, and case. Modifiers of a noun include adjectives, pronouns, cardinals, ordinals, some forms of verbs etc.

Subject and verb agreement
In Punjabi text, the verb must agree with the subject of the sentence in terms of gender, number, and person. There are some special forms of verbs like transitive past tense verbs, which need some specific postpositions with their subject, like the use of ਨੇ ne with transitive verbs in perfect form etc.

Noun and adjective (in attributive form) agreement
This is different from 'modifier and noun agreement' as described above in the sense that adjective is not preceding noun but can be virtually anywhere in the sentence, usually preceding verb phrase acting as a complement for it. It must still agree with the noun for which it is used in that sentence.

Order of the modifiers of a noun in noun phrase
If a noun has more than one modifier, then those modifiers should be in a certain order such that phrase modifiers precede single word modifiers but pronouns and numerals precede all other.

Order of the words in a verb phrase
There are certain future tense forms of Punjabi verbs that should occur towards the end of verb phrase without any auxiliary. In addition, if negative and emphatic particles are used in a verb phrase then the latter must precede the former.

ਦਾ da postposition and following noun phrase agreement
All the forms of ਦਾ da postposition must agree in terms of gender, number, and case with the following noun phrase that it is connecting with the preceding noun phrase.

Some other options covered include noun phrase must be in oblique form before a postposition, all the noun phrases joined by connectives must have same case, main verb should be in root form if preceding ਕੇ ke etc.

4 Sample Input and Output

This section provides some sample Punjabi sentences that were given as input to the Punjabi grammar checking system along with the output generated by this system.

Sentence 1
Shows the grammatical errors related to 'Modifier and noun agreement' and 'Order of the modifiers of a noun in noun phrase'. In this sentence noun is ਲੜਕਾ larka 'boy' and its modifiers are ਸੋਹਣੀ ਇੱਕ ਭੱਜੀ ਜਾਂਦਾ sohni ek bhajji janda 'handsome one running'.
Input: ਸੋਹਣੀ ਇੱਕ ਭੱਜੀ ਜਾਂਦਾ ਲੜਕਾ ਆਇਆ
Input1: sohni ek bhajji janda larka aaeya
Input2: Handsome one running boy came
Output: ਇੱਕ ਭੱਜਿਆ ਜਾਂਦਾ ਸੋਹਣਾ ਲੜਕਾ ਆਇਆ
Output1: ek bhajjia janda sohna larka aaeya
Output2: One running handsome boy came

Sentence 2
Covers the grammatical error related to 'Subject and verb agreement'. Subject is ਬਾਰਸ਼ barish 'rain' and verb phrase is ਹੋ ਰਿਹਾ ਹਨ ho riha han 'is raining'.
Input: ਬਾਹਰ ਬਾਰਸ਼ ਹੋ ਰਿਹਾ ਹਨ
Input1: bahr barish ho riha han
Input2: It is raining outside
Output: ਬਾਹਰ ਬਾਰਸ਼ ਹੋ ਰਹੀ ਹੈ
Output1: bahr barish ho rahi hai
Output2: It is raining outside

Sentence 3
For grammatical errors related to 'ਦਾ da postposition and following noun phrase agreement' and 'Noun phrase in oblique form before a post position'. Noun phrase preceding ਦੀ dee

(possessive marker) is ਛੋਟਾ ਬੱਚਾ chota baccha 'small boy' and following one is ਨਾਮ naam 'name'.

Input: ਛੋਟਾ ਬੱਚਾ ਦੀ ਨਾਮ ਰਾਮ ਹੈ

Input1: chota baccha dee naam raam hai
Input2: Small boy's name is Ram
Output: ਛੋਟੇ ਬੱਚੇ ਦਾ ਨਾਮ ਰਾਮ ਹੈ

Ouput1: chote bacche da naam raam hai
Ouput2: Small boy's name is Ram

Sentence 4
Highlights the grammatical errors related to 'Subject and verb agreement' and 'Order of the words in a verb phrase'. Subject in this sentence is ਲੜਕੀ larki 'girl' and verb phrase is ਨਹੀਂ ਜਾ ਹੀ ਰਿਹਾ ਸੀ nahi ja hee riha see 'was not going'.

Input: ਲੜਕੀ ਸਕੂਲ ਨਹੀਂ ਜਾ ਹੀ ਰਿਹਾ ਸੀ

Input1: larkee school nahi ja hee riha see
Input2: The girl was not going to school
Output: ਲੜਕੀ ਸਕੂਲ ਜਾ ਹੀ ਨਹੀਂ ਰਹੀ ਸੀ

Ouput1: larkee school ja he nahi rahi see
Output2: The girl was not going to school

Sentence 5
For grammatical error related to 'Subject and verb agreement'. Subject here is ਰਾਮ raam 'Ram' and verb phrase is ਖਾਧਾ khadha 'ate', which is transitive and in perfect phase.
Input: ਰਾਮ ਫ਼ਲ ਖਾਧਾ

Input1: raam phal khadha
Input2: Ram ate fruit
Output: ਰਾਮ ਨੇ ਫ਼ਲ ਖਾਧਾ

Output1: raam ne phal khadha
Output2: Ram ate fruit

Legend:

- **Input** and **Output** specifies the input Punjabi sentence in Gurmukhi script and the output produced by this grammar checking system in Gurmukhi script, respectively.
- **Input1/Output1** specifies the Romanized version of the **input/output**.
- **Input2/Output2** specifies the English gloss for the **input/output**.

5 System Features

The system is designed in Microsoft Visual C# 2005 using Microsoft .NET Framework 2.0. The entire database of this tool is in XML files with the Punjabi text in Unicode format. Some of the significant features of this grammar checking system are:

Rules can be turned on and off individually
Being a rule-based system all the rules provided in section 3 can be turned on and off individually without requiring any changes in the system. The rules are kept in a separate XML file, not hard coded into the system. To turn on/off a rule, changes can be made to that XML file directly or it can be done through the options provided within the system.

Error and Suggestions information
The system is able to provide enough reasons in support of every error that it detects. With a meaningful description of the rule, it provides the grammatical categories that failed to match if there is an error and provides the desired correct value for those grammatical categories, with suggestions. However, the information about grammatical categories may not be much meaningful to an ordinary user but if someone is learning Punjabi as a foreign/second language then information about correct grammatical categories according to the context can be helpful. Wherever possible system also specifies both the words, for which matching was performed, making it more clear that what is wrong and with respect to what, as shown in Figure 1, it shows that which was the head word and which word failed to match with it.

The suggestions produced by the Punjabi Grammar Checker for the following grammatically incorrect sentence to correct the first incorrect word ਰਹੀਆਂ rahian '-ing plural' are ਰਿਹਾ riha '-ing singular' and ਰਹੀ rahi '-ing singular':

ਮੈਂ ਖੇਡ ਰਹੀਆਂ ਹਨ

main khed rahian han 'I are playing'

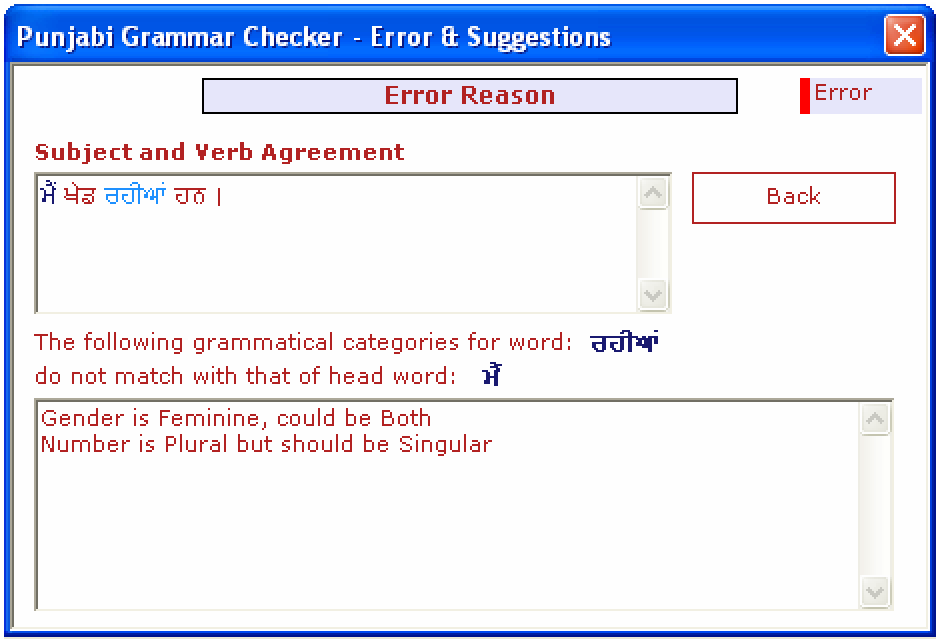


Figure 1. Punjabi Grammar Checker – Error Reason

Figure 1 shows the grammatical categories that failed to match for the subject ਮੈਂ main 'I' and part of the verb phrase ਰਹੀਆਂ rahian '-ing plural'. It provides the values for the grammatical categories that failed to match for the incorrect word along with the desired values for correction.

6 System Scope

The system is designed to work on the literary style Punjabi text with SOV (Subject Object Verb) sentence structure. At present, it works properly on simple or kernel sentences. It can detect any agreement errors in compound or complex sentences also. However, there may be some false alarms in such sentences. The sentences in which word order is shuffled for emphasis has not been considered, along with the sentences in which intonation alone is used for emphasis. Due to emphatic intonation, the meaning or word class of a word may be changed in a sentence e.g., ਤੇ te 'and' is usually a connective but if emphasized it can be used as an emphatic particle. However, this is hard to detect from the written form of the text and thus has not been considered. However, if some emphatic particles like ਹੀ he ਈ ee ਵੀ ve etc., are used directly in a sentence to show emphasis then that is given due consideration.

7 Hardware and Software Requirements

The system needs hardware and software as would be expected from a typical word processing application. A Unicode compatible Windows XP based PC with 512 MB of RAM, 1 GB of hard disk space and Microsoft .NET Framework 2.0 installed, would be sufficient.

References

Duni Chander. 1964. *Punjabi Bhasha da Viakaran (Punjabi).* Punjab University Publication Bureau, Chandigarh, India.

Daniel Naber. 2003. *A Rule-Based Style and Grammar Checker.* Diplomarbeit Technische Fakultät, Universität Bielefeld, Germany. (Available at: http://www.danielnaber.de/languagetool/download/style_and_grammar_checker.pdf (1/10/2007))

Harjeet S. Gill and Henry A. Gleason, Jr. 1986. *A Reference Grammar of Punjabi*. Publication Bureau, Punjabi University, Patiala, India.

Joginder S. Puar. 1990. *The Punjabi verb form and function*. Publication Bureau, Punjabi University, Patiala, India.

Md. Jahangir Alam, Naushad UzZaman, and Mumit Khan. 2006. N-gram based Statistical Grammar Checker for Bangla and English. In *Proc. of ninth International Conference on Computer and Information Technology (ICCIT 2006)*, Dhaka, Bangladesh.

Netgraph – Making Searching in Treebanks Easy

Jiří Mírovský
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské nám. 25, 118 00 Prague 1, Czech Republic
`mirovsky@ufal.mff.cuni.cz`

1 Introduction

Searching in a linguistically annotated treebank is a principal task that requires a sophisticated tool. Netgraph has been designed to perform the searching with maximum comfort and minimum requirements on its users. Although it has been developed primarily for the Prague Dependency Treebank 2.0 (Hajič et al. 2006), it can be used with other treebanks too, both dependency and constituent-structure types, as long as the treebank is transformed to a suitable format.

In this paper, we present Netgraph query language and on many examples show how it can be used to search for frequent linguistic phenomena.

In *section 1* (after this introduction) we extremely briefly describe the Prague Dependency Treebank 2.0, just to make the examples in the subsequent text more understandable. In the next subsection we mention the history of Netgraph and its properties as a tool.

In *section 2* we offer an introduction to the query language of Netgraph along with the idea of meta-attributes and what they are good for, and present linguistically motivated examples of queries in the Prague Dependency Treebank.

Finally, in *section 3* we offer some concluding remarks.

1.1 Prague Dependency Treebank 2.0

The Prague Dependency Treebank 2.0 (PDT 2.0, see Hajič et al. 2006, Hajič 2004) is a manually annotated corpus of Czech. It is a sequel to the Prague Dependency Treebank 1.0 (PDT 1.0, see Hajič et al. 2001a, Hajič et al. 2001b).

The texts in PDT 2.0 are annotated on three layers - the morphological layer, the analytical layer and the tectogrammatical layer. The corpus size is almost 2 million tokens (115 thousand sentences), although “only” 0.8 million tokens (49 thousand sentences) are annotated on all three layers. By 'tokens' we mean word forms, including numbers and punctuation marks.

On the morphological layer (Hana et al. 2005), each token of every sentence is annotated with a lemma (attribute `m/lemma`), keeping the base form of the token, and a tag (attribute `m/tag`), keeping its morphological information. Sentence boundaries are annotated here, too.

The analytical layer roughly corresponds to the surface syntax of the sentence; the annotation is a single-rooted dependency tree with labeled nodes (Hajič et al. 1997, Hajič 1998). Attribute `afun` describes the type of dependency between a dependent node and its governor. The nodes on the analytical layer (except for technical roots of the trees) also correspond 1:1 to the tokens of the sentences. The order of the nodes from left to right corresponds exactly to the surface order of tokens in the sentence (attribute `ord`). Non-projective constructions (that are quite frequent both in Czech (Hajičová et al. 2004) and in some other languages (see Havelka 2007)) are allowed.

The tectogrammatical layer captures the linguistic meaning of the sentence in its context. Again, the annotation is a dependency tree with labeled nodes (see Hajičová 1998). The correspondence of the nodes to the lower layers is more complex here. It is often not 1:1, it can be both 1:N and N:1. It was shown in detail in Mírovský (2006) how Netgraph deals with this issue.

Attribute `functor` describes the dependency between a dependent node and its governor. A tec-

togrammatical lemma (attribute t_lemma) is assigned to every node. Grammatemes, which keep additional annotation, are rendered as a set of 16 attributes grouped by the "prefix" gram (e.g. gram/verbmod for verbal modality).

Topic and focus (Hajičová et al. 1998) are marked (attribute tfa), together with so-called deep word order reflected by the order of nodes in the annotation (attribute deepord).

Coreference relations between nodes of certain category types are captured (Kučová et al. 2003), distinguishing also the type of the relation (textual or grammatical). Each node has an identifier (attribute id) that is unique throughout the whole corpus. Attributes coref_text.rf and coref_gram.rf contain ids of coreferential nodes of the respective types.

1.2 Netgraph as a Tool

The development of Netgraph started in 1998 as a topic of Ondruška's Master's thesis (Ondruška 1998), and has been proceeding along with the ongoing annotations of the Prague Dependency Treebank 1.0 and later the Prague Dependency Treebank 2.0. Now it is a fully functional tool for complex searching in PDT 2.0 and other treebanks.

Netgraph is a client-server application that allows multiple users to search the treebank on-line and simultaneously through the Internet. The server (written in C) searches the treebank, which is located at the same computer or local network. The client (written in Java2) serves as a very comfortable graphical user interface and can be located at any node in the Internet. The client exists in two forms: as an applet and as a stand-alone application. The applet version can be run from Netgraph home page and searches in PDT 2.0. The stand-alone version can be downloaded from the same page and can connect anonymously to PDT 2.0 server. More information can be found on Netgraph home page (http://quest.ms.mff.cuni.cz/netgraph).

The client sends user queries to the server and receives results from it. Both the server and the client also can, of course, reside at the same computer. Authentication by the means of login names and passwords is provided. Users can have various access permissions.

A detailed description of the inner architecture of Netgraph and of the communication between the server and the client was given in Mírovský, Ondruška and Průša (2002).

Netgraph server requires the treebank in FS format, encoded in UTF-8. A formal description of the format can be found in Hajič et al. 2001a. Netgraph query language, presented in the next section, is an extension of FS format.

2 Netgraph Query Language

In this section we give an introduction to Netgraph query language. We show on a series of examples how some frequent linguistic phenomena can be searched for.

2.1 The Query Is a Tree

The query in Netgraph is a tree that forms a subtree in the result trees. The treebank is searched tree by tree and whenever the query is found as a subtree of a tree (we say the query and the tree match), the tree becomes part of the result. The result is displayed tree by tree on demand. The query can also consist of several trees joined either by AND or OR relation. In that case, all the query trees at the same time (or at least one of the query trees, respectively) are required to match the result tree.

The query has both a textual form and a graphical form. In the following text, we will use its textual form for simple queries and its graphical form (or both forms) for more advanced queries.

The syntax of the language is very simple. In the textual form, square brackets enclose a node, attributes (pairs name=value) are separated by a comma, quotation marks enclose a regular expression in a value. Parentheses enclose a subtree of a node, brothers are separated by a comma. In multiple-tree queries, each tree is on a new line and the first line contains only a single AND or OR. Alternative values of an attribute, as well as alternative nodes, are separated by a vertical bar. It almost completes the description of the syntax, only one thing (references) will be added in the following subsection.

The simplest possible query (and probably of little interest on itself) is a simple node without any evaluation: []. It matches all nodes of all trees in the treebank, each tree as many times as how many nodes there are in the tree. Nevertheless, we may add conditions on its attributes, optionally using regular expressions in values of the attributes. Thus

we may search e.g. for all nodes that are Subjects and nouns but not in first case:

```
[afun=Sb, m/tag="N...[^1].*"].
```

We may notice here that regular expressions allow the first (very basic) type of negation in queries.

More interesting queries usually consist of several nodes, forming a tree structure. The following example query searches for trees containing a Predicate that directly governs a Subject and an Object:

```
[afun=Pred]([afun=Sb],[afun=Obj]).
```

Please note that there is no condition in the query on the order of the Subject and the Object, nor on their left-right position to their father. It does not prevent other nodes to be directly governed by the Predicate either.

2.2 Meta-Attributes

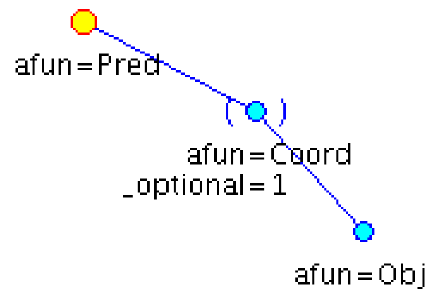
This simple query language, described briefly in only a few examples, is quite useful but not powerful enough. There is no possibility to set a real negation, no way of restricting the position of the query in the result tree or the size of the result tree, nor the order of nodes can be controlled. To allow these and other things, meta-attributes have been added to the query system.

Meta-attributes are not present in the corpus but they pretend to be ordinary attributes and the user uses them the same way like normal attributes. Their names start with an underscore. There are eleven meta-attributes, each adding some power to the query language, enhancing its semantics, while keeping the syntax of the language on the same simple level. We present several of the meta-attributes in this subsection, some others will be presented in the subsequent section, when they are needed. A detailed description of the principal meta-attributes was given in Mírovský (2006).

Coordination is a frequent phenomenon in languages. In PDT (and in most other treebanks, too) it is represented by a coordinating node. To be able to skip (and effectively ignore) the coordination in the queries, we have introduced the meta-attribute `_optional` that marks an optional node. The node then may but does not have to be in the result. If we are interested, for example, in Predicates governing Objects, we can get both cases (with coordination and without it) in one query using this meta-attribute:

```
[afun=Pred]([afun=Coord,_optional=1]([afun=Obj])).
```

The Coordination becomes optional. If there is a node between the Predicate and its Object in the result tree, it has to be the Coordination. But the Object may also be a direct son of the Predicate, omitting the optional Coordination. The picture demonstrates that the graphical representation of the query is much more comprehensible than its textual version:

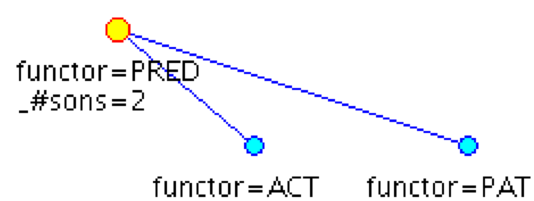


There is a group of meta-attributes of rather technical nature, which allow setting a position of the query in the result tree, restricting the size of the result tree or its part, and restricting number of direct sons of a node. Meta attribute `_depth` controls the distance of a node from the root (useful when searching for a phenomenon in subordinated clauses, for example), `_#descendants` controls number of nodes in the subtree of a node (useful e.g. when searching for „nice“ small examples of something), `_#sons` controls number of (direct) sons of a node.

Controlling number of direct sons (mainly in its negative sense) is important for studying valency of words (Hajičová and Panevová 1984). The following example searches on the tectogrammatical layer of PDT. We want a Predicate that governs directly an Actor and a Patient and nothing else (directly):

```
[functor=PRED,_#sons=2]([functor=ACT],[functor=PAT]).
```

The graphical representation of the query is:



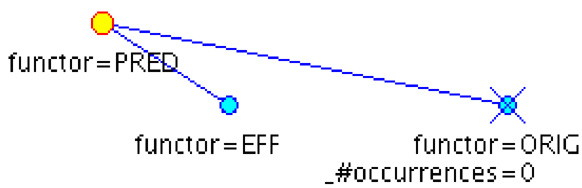
If we replaced PAT with ADDR, we might search for errors in the evaluation, since the theory

forbids Actor and Addressee being the only parts of a valency frame.

So far, we could only restrict number of nodes. But we often want to restrict a presence of a certain type of node. We want to specify that there is not a node of a certain quality. For example, we might want to search (again on the tectogrammatical layer) for an Effect without an Origo in a valency frame. The meta-attribute that allows this real type of negation is called `_#occurrences`. It controls the exact number of occurrences of a certain type of node, in our example of Origos:

```
[functor=PRED]([functor=EFF],
[functor=ORIG,_#occurrences=0])
```

with graphical representation:



It says that the Predicate has at least one son – an Effect, and that the Predicate does not have an Origo son.

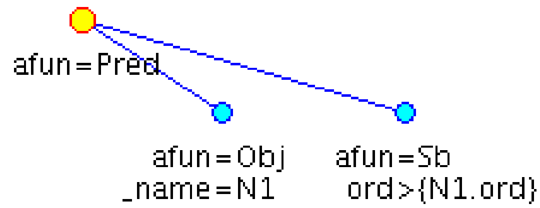
There is still one important thing that we cannot achieve with the meta-attributes presented so far. We cannot set any relation (other than dependency) between nodes in the result trees (such as order, agreement in case, coreference). All this can be done using the meta-attribute `_name` and a system of references. The meta-attribute `_name` simply names a node for a later reference from other nodes.

Curly brackets enclose a reference to a value of an attribute of the other node (with a given name) in the result tree. This, along with the dot-referencing inside the reference and some arithmetic possibilities, completes our description of the syntax of the query language from subsection 2.1.

In the following example (back on the analytical layer and knowing that attribute `ord` keeps the order of the nodes (~ tokens) in the tree (~ sentence)) from left to right, we search for a Subject that is on the right side from an Object (in the tree and also in the sentence):

```
[afun=Pred]
([afun=Sb,ord>{N1.ord}],
[afun=Obj,_name=N1])
```

with graphical representation:



We have named the Object node `N1` and specified that ord of the Subject node should be bigger than `ord` of the `N1` node. If we used `ord>{N1.ord}+5`, we would require them to be at least five words apart.

Meta-attribute _#lbrothers (_#rbrothers) contains number of left (right) brothers of a particular node in the result tree. Thus, we can define that a node (e.g. an Attribute) is the leftmost son of another node (e.g. an Object):

```
[afun=Obj]([afun=Atr,_#lbroth-
ers=0]).
```

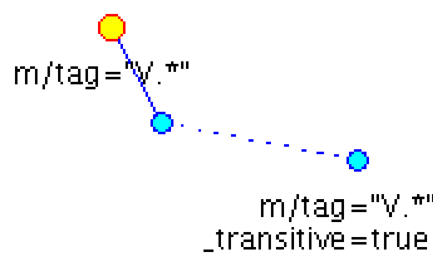
Meta-attribute `_transitive` defines a transitive edge. The following example searches for a verb node that governs transitively another verb node:

```
[m/tag="V.*"]([m/tag="V.*",_tran-
sitive=true]).
```

If we do not want them to be direct father and son, we have two possibilities: Either we put another node without any evaluation in between them in the query:

```
[m/tag="V.*"]([]([m/tag="V.*",
_transitive=true]))
```

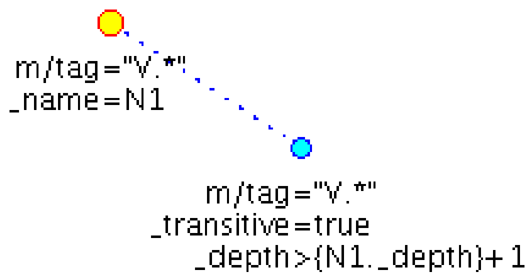
with graphical representation:



or, we can use meta-attribute `_depth` and references:

```
[m/tag="V.*",_name=N1]
([m/tag="V.*",_transitive=true,
_depth>{N1._depth}+1])
```

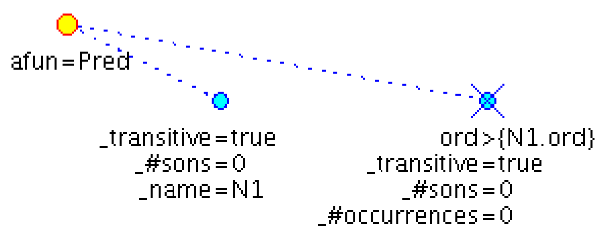
which is perhaps a little bit more complex. The graphical representation of the query is:



Using several meta-attributes in one query can form a powerful combination. The following example searches for the rightmost list descendant of a Predicate:

```
[afun=Pred]([_transitive=true,
_#sons=0,_name=N1],[_transi-
tive=true,_#sons=0,
ord>{N1.ord},_#occurrences=0])
```

with graphical representation:



The first transitive descendant of the Predicate is the list (`_#sons=0`) we are looking for. The second transitive descendant is a list that we do not want to be in the result (with higher `ord`). Therefore, we set `_#occurrences` to zero.

3 Conclusion

We have presented Netgraph query language, its basics and also its advanced techniques, namely meta-attributes, references and their combination.

We have demonstrated that many linguistic phenomena can be searched for using this language. It can be shown (Mírovský 2008) that Netgraph querying power outperforms the querying power of TGrep (Pito 1994), which is a traditional (and nowadays outdated) treebank searching tool. On the other hand, it seems (it has not been studied thoroughly yet) that Netgraph has slightly lesser searching power than TGrep2 (Rohde 2005), which can use any boolean combination of its searching patterns.

Acknowledgement The research reported in this paper was supported by the Grant Agency of the Academy of Sciences of the Czech Republic, project IS-REST (No. 1ET101120413).

References

Hajič J. et al. 2006. Prague Dependency Treebank 2.0. *CD-ROM LDC2006T01, LDC, Philadelphia, 2006.*

Hajič J. 2004. Complex Corpus Annotation: The Prague Dependency Treebank. *Jazykovedný ústav Ľ. Štúra, SAV, Bratislava, 2004.*

Hajič J., Vidová-Hladká B., Panevová J., Hajičová E., Sgall P., Pajas P. 2001a. Prague Dependency Treebank 1.0 (Final Production Label). *CD-ROM LDC2001T10, LDC, Philadelphia, 2001.*

Hajič J., Pajas P. and Vidová-Hladká B. 2001b. The Prague Dependency Treebank: Annotation Structure and Support. *In IRCS Workshop on Linguistic databases, 2001, pp. 105-114.*

Hana J., Zeman D., Hajič J., Hanová H., Hladká B., Jeřábek E. 2005. Manual for Morphological Annotation, Revision for PDT 2.0. *ÚFAL Technical Report TR-2005-27, Charles University in Prague, 2005.*

Hajič J. et al. 1997. A Manual for Analytic Layer Tagging of the Prague Dependency Treebank. *ÚFAL Technical Report TR-1997-03, Charles University in Prague, 1997.*

Hajič J. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. *In Issues of Valency and Meaning, Karolinum, Praha 1998, pp. 106-132.*

Hajičová E., Havelka J., Sgall P., Veselá K., Zeman D. 2004. Issues of Projectivity in the Prague Dependency Treebank. *MFF UK, Prague, 81, 2004.*

Havelka J. 2007. Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures. *In Proceedings of ACL 2007, Prague, pp. 608-615.*

Hajičová E. 1998. Prague Dependency Treebank: From analytic to tectogrammatical annotations. *In: Proceedings of 2nd TST, Brno, Springer-Verlag Berlin Heidelberg New York, 1998, pp. 45-50.*

Hajičová E, Panevová J. 1984. Valency (case) frames. *In P. Sgall (ed.): Contributions to Functional Syntax, Semantics and Language Comprehension, Prague, Academia, 1984, pp. 147-188.*

Mírovský J. 2006. Netgraph: a Tool for Searching in Prague Dependency Treebank 2.0. *In Proceedings of TLT 2006, Prague, pp. 211-222.*

Hajičová E., Partee B., Sgall P. 1998. Topic-Focus Articulation, Tripartite Structures and Semantic Content. *Dordrecht, Amsterdam, Kluwer Academic Publishers, 1998.*

Kučová L., Kolářová-Řezníčková V., Žabokrtský Z., Pajas P., Čulo O. 2003. Anotování koreference v Pražském závislostním korpusu. *ÚFAL Technical Report TR-2003-19, Charles University in Prague, 2003.*

Ondruška R. 1998. Tools for Searching in Syntactically Annotated Corpora. *Master Thesis, Charles University in Prague, 1998.*

Mírovský J., Ondruška R., Průša D. 2002. Searching through Prague Dependency Treebank - Conception and Architecture. *In Proceedings of The First Workshop on Treebanks and Linguistic Theories, Sozopol, 2002, pp. 114--122.*

Mírovský J.: Netgraph Home Page: *http://quest.ms.mff.-cuni.cz/netgraph*

Mírovský J. 2008. Towards a Simple and Full-Featured Treebank Query Language. *In Proceedings of First International Conference on Global Interoperability for Language Resources, Hong Kong, 2008, in print.*

Pito R. 1994. TGrep Manual Page. *Available from http://www.ldc.upenn.edu/ldc/online/treebank/*

Rohde D. 2005. TGrep2 User Manual. *Available from http://www-cgi.cs.cmu.edu/~dr/TGrep2/tgrep2.pdf*

Global Health Monitor - A Web-based System for Detecting and Mapping Infectious Diseases

Son Doan*, QuocHung-Ngo¶, Ai Kawazoe*, Nigel Collier*

* National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo,
Japan
{doan,zoeai,collier}@nii.ac.jp

¶ University of Information Technology,
Vietnam National University (HCM),
Vietnam
hungnq@uit.edu.vn

Abstract

We present the Global Health Monitor, an online Web-based system for detecting and mapping infectious disease outbreaks that appear in news stories. The system analyzes English news stories from news feed providers, classifies them for topical relevance and plots them onto a Google map using geo-coding information, helping public health workers to monitor the spread of diseases in a geo-temporal context. The background knowledge for the system is contained in the BioCaster ontology (BCO) (Collier et al., 2007a) which includes both information on infectious diseases as well as geographical locations with their latitudes/longitudes. The system consists of four main stages: topic classification, named entity recognition (NER), disease/location detection and visualization. Evaluation of the system shows that it achieved high accuracy on a gold standard corpus. The system is now in practical use. Running on a cluster-computer, it monitors more than 1500 news feeds 24/7, updating the map every hour.

1 Introduction

Information concerning disease outbreak events is published in various news outlets on the World Wide Web, in many different languages. Identifying early news stories about disease outbreaks *automatically* is important for a bio-surveillance system that is designed to inform health professionals. Currently, there are several systems available for the disease detection and tracking task. For example, ProMED-mail (2001) or MedISys (2007) (Medical Intelligence System). ProMED-mail is an Internet-based system that provides reports by public health experts concerning outbreak diseases (that is, the system is *not* automatic but rather human curated). In contrast to ProMED-mail, MedISys is an automatic system working on multilingual languages, but it mainly focuses on analyzing news stories based on the country level. Another system which is close to the one we present is HealthMap (Brownstein and Freifeld, 2007). HealthMap automatically collects news from the Internet about human and animal health and plots the data on a Google Maps mashup. Data is aggregated by disease and location. Unlike HealthMap, our system takes an ontology-centred approach to knowledge understanding and linkage to external resources. For annotation of topics and entities we also exploit a range of linguistic resources within a machine learning framework.

There are several challenges in geo-coding when dealing with news stories. The two main challenges are disease/location extraction and geo-disambiguation. The former is concerned with how to determine disease and location names for disease-related news stories. The latter is concerned with how to solve geo-disambiguation. For example, if there is a news story about equine influenza in Camden, the system should detect that the disease name is “equine influenza” and the location name is “Camden”. However, there are two locations named Camden: One in Australia and one in

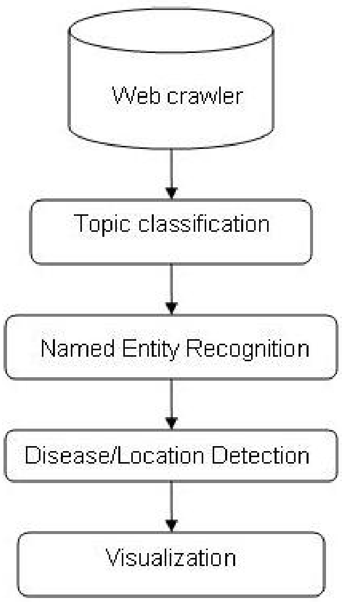


Figure 1. Stages of the system.

London, UK. The problem is that only one location should be chosen for plotting into a map. In our opinion, current systems lack the advanced natural language processing and text mining techniques that would enable the automatic extraction of such disease/location event information.

BioCaster is a project working towards the detection and tracking of infectious diseases using text mining techniques. One of the main components is the BioCaster Ontology (BCO), which includes 50 infectious diseases with links to external vocabulary resources, and a geographical ontology of a total of 243 countries and 4025 sub-countries (province and cities) with their latitudes/longitudes. We also now automatically link news on outbreaks to academic sources such as Stanford university's Highwire and NCBI's PubMed using search terms Disease name + Location name (Country) + "case". This is to ensure a focus on recent case report relevant to the news items.

The system includes four main stages: topic classification, named entity recognition (NER), disease/location detection, and visualization. The current version of the system (English only) can be found at http://biocaster.nii.ac.jp.

The remainder of this paper is organized as follows. Section 2 outlines the BioCaster Ontology (BCO). Section 3 describes some features of the system (modules, functionality and algorithms). Section 4 is concerned with system evaluation. Finally, Section 5 outlines conclusions and presents possible future work.

2 Overview of BCO

BCO is one of the main components in the BioCaster project. It includes an ontology of 50 infectious diseases and a geographical ontology (243 countries and 4,025 sub-countries). The infectious disease ontology was built by a team consisting of a linguist, an epidemiologist, a geneticist, and a medical anthropologist. A disease in BCO has a root name which is unique identifier and also other properties relating to synonyms, symptoms, associated syndromes, hosts, etc. The ontology is multilingual, supporting six languages (English, Japanese, Vietnamese, Thai, Korean, and Chinese); and has links to external ontologies (such as MeSH, SNOMED and ICD9/10) and resources (like Wikipedia). The geographical part is built from Wikipedia[1]. The BCO is available on the Web at http://biocaster.nii.ac.jp. For a fuller description of the BCO, see Collier et al. (2007a) and Kawazoe et al. (2007).

3 The System

3.1 Overview of the system

The Global Health Monitor system runs on a cluster machine with 16 nodes under the Linux operating system. The code was written in PHP, Perl, C, and Java and the number of input news feeds is about 1,500. The system has a crawler that collects news every hour. These collected news stories are then processed and analyzed step-by-step in four main phases: topic classification, named entity recognition (NER), disease/location detection, and visualization. Each of the four phases is managed by a distinct module. These components are depicted in Figure 1. The first three modules are run inside the system and the visualization module – the Google Map – can be seen at the BioCaster portal. Figure 2 shows a screenshot of the system.

[1] http://www.wikipedia.org.

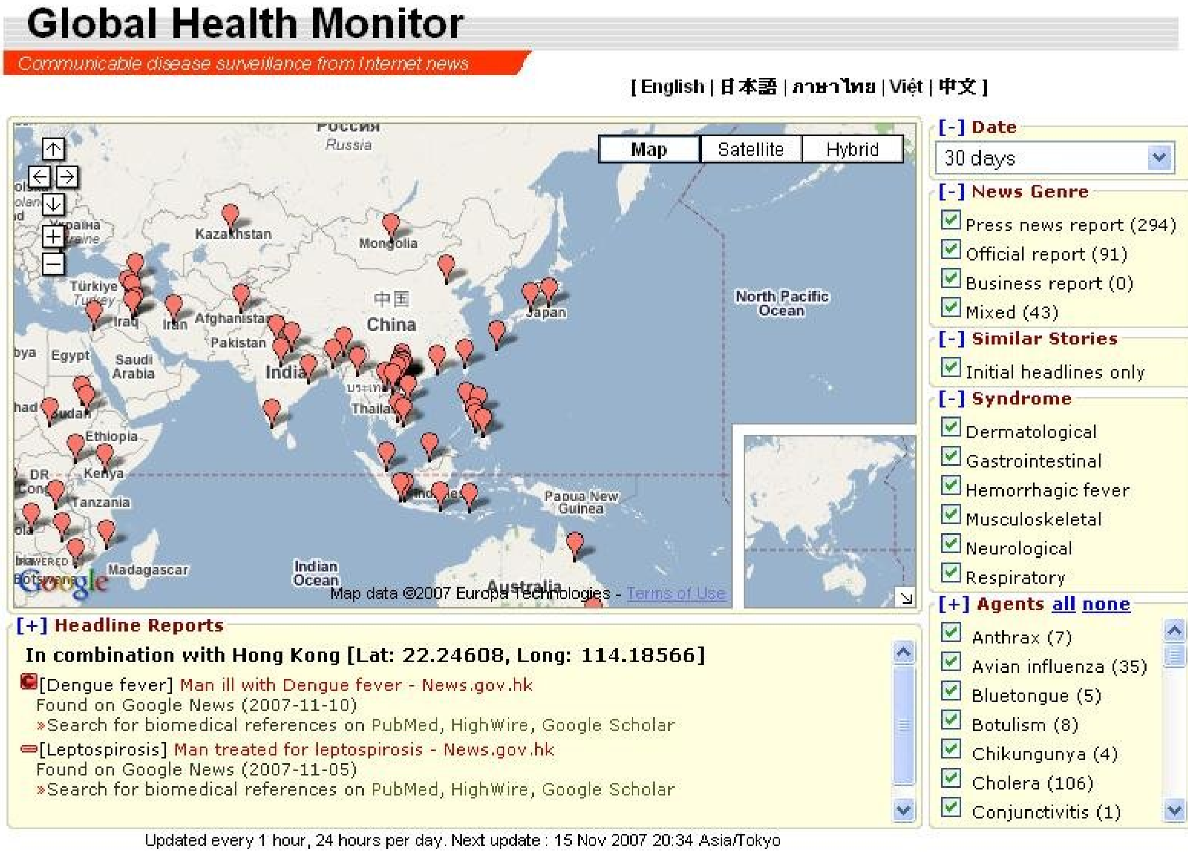


Figure 2. The Global Health Monitor system, showing disease events from the last 30 days. The main screen is a Google Map. Selected headline reports run along the bottom of the screen and link to biomedical reference on PubMed, HighWire and Google Scholar. Symbol links to disease names in the BCO and symbol stands for disease name not in the BCO. The right of the screen shows various user options to filter the news.

We will now describe each of the four modules in turn:

*** Topic classification.** This module identifies news stories with disease-related topics and retains relevant ones for later processing. The module uses ontology-supported text classification with naïve Bayes as the classification algorithm and the BioCaster gold standard corpus as the training data set (Doan et al., 2007). In this module, we used the Rainbow toolkit.[2]

*** NER.** Disease-related news stories are automatically analyzed and tagged with NEs like PERSON, ORGANIZATION, DISEASE, LOCATION. This module is implemented by SVM classification algorithm[3]. For a more detailed description of the schema and NER module, see Kawazoe et al. (2006).

*** Disease/location detection.** This module extracts disease and location information. Details are given in Section 3.2.

*** Visualization.** The detected locations are plotted onto a Google map with ontology links to associated diseases and news stories.

3.2 Disease/location detection algorithm

The disease/location detection algorithm is based on a statistical model of the LOCATION and DIS EASE Named Entities (NEs). The algorithm can be described as follows:

[2] Rainbow toolkit, available at http://www.cs.umass.edu/~mccallum/bow/rainbow

[3] TinySVM, available at http://chasen.org/~taku/software/TinySVM.

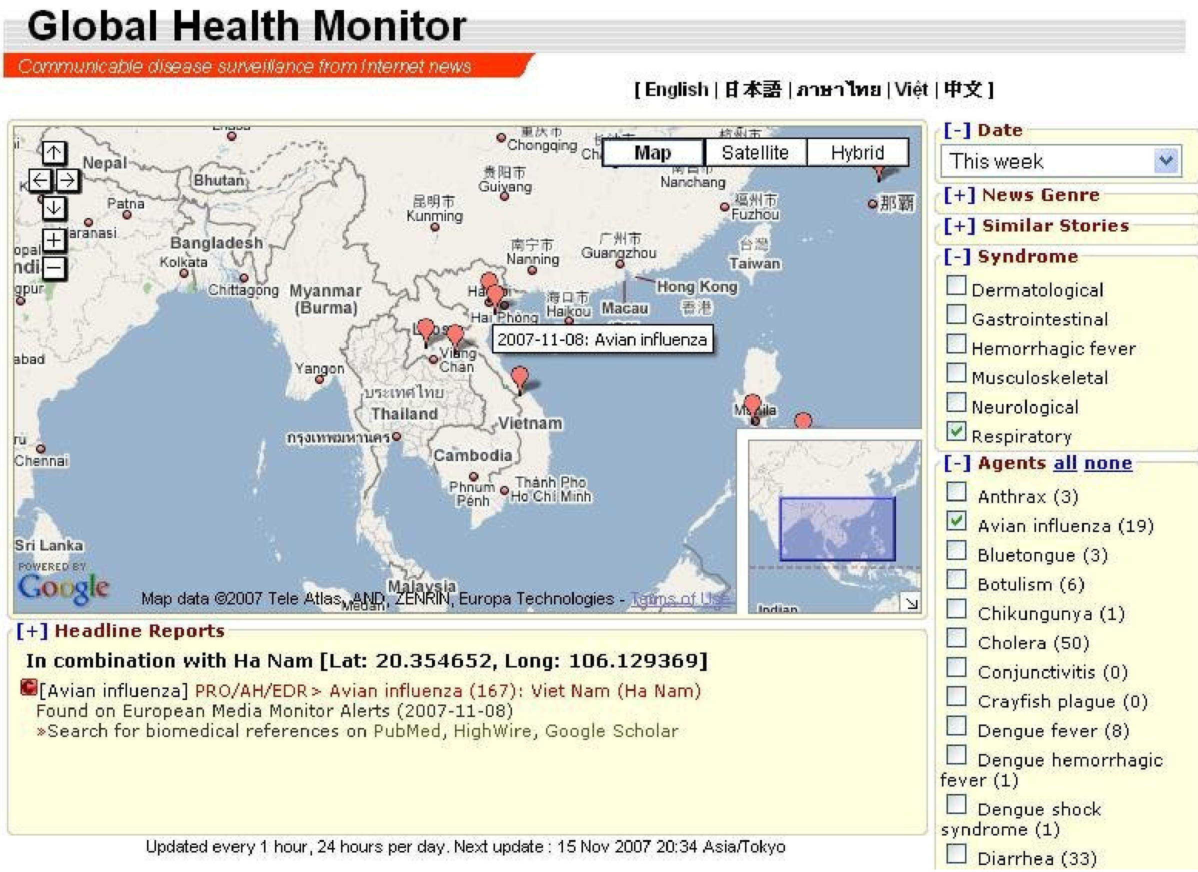


Figure 3. The Global Health Monitor with the *Respiratory Syndrome* selected. The time span selected is the current week.

Input: A set of news stories tagged with NEs.
Output: A set of disease/location pairs.

Step 1: Detect LOCATION-DISEASE pairs in each news story by corresponding NEs, and calculate their frequency in a news story.

Step 2: Calculate the frequency of LOCATION-DISEASE pairs in a corpus.

Step 3: Rank LOCATION – DISEASE pairs by the frequencies calculated in Step 2. Use a threshold to choose top LOCATION - DISEASE names[4].

Step 4: Map disease and location names: If DISEASE matches to a synonym in BCO then DISEASE was assigned to that disease name. This process of matching (grounding the terms) allows us to provide extra information from the ontology and to remove variant forms of terms from the map – thereby aiding readability. Similarly, if LOCATION matches to a location in BCO then LOCATION was assigned to that location name.

Step 5: Re-map into news stories: Match detected diseases and locations within the first half of each news story. If both disease and location are matched then they are stored; otherwise, skip.

This five step process is repeated every hour, for each news article that is less than 1 day (24 hours) old.

3.3 Capabilities of the system

The following lists some capabilities of the current Global Health Monitor system.

* Date range: The system shows the disease/location and news stories within a specific date range. Current implemented date ranges are: 30 days ago, 3 weeks ago, 2 weeks ago, 1 week ago, this week and today.

[4] In the current system, we set the threshold value to 40.

* Genre filter: The system can show news stories by publication type. There are four genres of news: Press news (like Google News, Yahoo News), Official news (like ProMED, WHO reports), Business news, and Mixed news (like Topix.com).

* Similar stories: The system currently uses a simple method to remove duplicate news stories. Users can use the “Initial headline only” option to activate this function.

* Syndrome filter: There are six syndromes in BCO: Dermatological, Gastrointestinal, Hemorharrgic fever, Musculoskeletal, Neurological, and Respiratory. A syndrome can be associated with several diseases included in BCO. The system can show news stories related to these syndromes.

* Agent option: This option allows users to view lists of infectious diseases which come from BCO. Some diseases though are not in the BCO. Users can choose some, all, or no diseases using a checkbox style interface.

Figure 3 shows the interface when users choose Syndromes as Respiratory for this week at the current view.

4 Evaluation

To evaluate any bio-surveillance system is very challenging (Morse, 2007). Our system is an integration of several modules, e.g., classification, NER and other algorithms. The evaluation processes for these modules are briefly described below:

4.1 Topic classification

Evaluation of topic classification is presented in Doan et al. (2007). The system used the BioCaster gold standard corpus which includes 1,000 annotated news stories as training data. The classification model is naïve Bayes with features as raw text, NEs, and Roles (Doan et al., 2007). The system achieved an accuracy score of 88.10%.

4.2 NER evaluation

The evaluation of the NER system module is reported in Kawazoe et al. (2006). We used an annotated corpus of 200 corpus news articles as training data. The NER system achieved an F-score of 76.97% for all NE classes.

4.3 Disease/location detection

For the preliminary evaluation of disease/location detection, we used data from a one-month period (from October 12 to November 11, 2007).

In our observations, the system detects about 25-30 locations a day, an average of 40 infectious diseases and 950 detected pairs of diseases/locations per month (A news story can contain multiple locations and diseases). The main news resources mostly come from Google News (251 pairs, about 26.4%), Yahoo News (288 pairs, about 30.3%), ProMED-mail (180 pairs, about 18.9%), and the remaining 24.3% for others. The running time for updating disease/location takes about 5 minutes.

In order to evaluate the performance of disease/location detection, we define the Precision and Recall as follows:

$$\text{Precision} = \frac{\#\text{Relevant pairs} \cap \#\text{Retrieved pairs}}{\#\text{Retrieved pairs}}$$

$$\text{Recall} = \frac{\#\text{Relevant pairs} \cap \#\text{Retrieved pairs}}{\#\text{Relevant pairs}},$$

Where #Relevant pairs is the number of disease/location pairs that human found, and #Retrieved pairs is the number of disease/location pairs that the system detected.

The Precision can be calculated based on our retrieved pairs detected by the system, however the Recall is under estimated as it does not measure pairs missed by the system in the topic classification stage.

We evaluate the Precision of disease/location detection on 950 pairs of location/disease. The system correctly detected 887 pairs, taking 887/950=93.4% Precision.

4.4 Limitations of the system

There are some limitations of the system. The first limitation is there are several cases of ambiguity. For example, news stories about “A team at Peking University in Beijing studied tissue taken from 2

people killed by H5N1 in China" or "A meeting on foot and mouth disease (FMD) was held in Brussels on 17th October, 2007". The system incorrectly detects the location as Beijing in the first story, and Brussels in the second one. Another hard case is location disambiguation, e.g., news about "Rabies in Isle of Wight" in which in the main body does not mention anything about country and sub-country. There are two locations named "Isle of Wight" in our geo-ontology: one in Virginia, USA and one in the UK. In the future, we will look at the country-level information of new providers (by checking domain names) to solve this problem. For example, if a news story mentions the Isle of Wight, and the news story originates from the UK, then it will be taken to refer to the Isle of Wight in the UK.

The second limitation is the ability to detect new diseases or locations that are not in the ontology. In the future work, we will augment newly detected diseases as well as improve the geographical ontology.

5 Conclusion

We presented the Global Health Monitor - a Web-based system for detecting and mapping infectious diseases from Web. The system collects news from news feed providers, analyzes news and plots disease relevant data onto a Google map. Preliminary evaluations show that our system works efficiently with real data.

In the future, we will develop more efficient algorithms for detecting diseases/locations based on relation identification. Named relation will be described in the BCO event taxonomy (Kawazoe et al., 2007). Extra capabilities will be added to the system like classifying outbreak of disease by countries, detecting new diseases that are not in out ontology, and showing timeline of news stories. Evaluation of the timelineness system against human curated sources like ProMED-mail will be implemented. Working versions for other languages like Vietnamese, Japanese, and Thai are also being considered, using the existing BioCaster disease ontology.

Acknowledgements

The authors wish to thank Mike Conway at the National Institute of Informatics for revising the manuscript, and both Mika Shigematsu and Kiyosu Taniguchi at the National Institute of Infectious Diseases for useful discussions. This work was supported by Grants-in-Aid from the Japan Society for the Promotion of Science (grant no. 18049071).

References

J. Brownstein and C. Freifeld. 2007. *HealthMap – Global Disease Alert Mapping System*. http://www.healthmap.org.

N. Collier, A. Kawazoe, L. Jin, M. Shigematsu, D. Dien, R. Barrero, K. Takeuchi, A. Kawtrakul. 2007a. *A multilingual ontology for infectious disease outbreak surveillance: rationale, design and challenges*. Journal of Language Resources and Evaluation. DOI: 10.1007/s10579-007-9019-7.

N. Collier, A.Kawazoe, S. Doan, M. Shigematsu, K. Taniguchi, L. Jin, J. McCrae, H. Chanlekha, D. Dien, Q. Hung, V.C. Nam, K. Takeuchi, A. Kawtrakul. 2007b. *Detecting Web Rumors with a Multilingual Ontology - Supported Text Classification System.* Advances in Disease Surveillance, pp.242, vol.4, 2007.

S. Doan, A. Kawazoe, and N.Collier. 2007. *The Roles of Roles in Classifying Annotated Biomedical Text*. Proc. of BioNLP - Biological, translational, and clinical language processing 2007, pp.17-24, 2007.

A. Kawazoe, L. Jin, M. Shigematsu, R. Barrero, K. Taniguchi and N. Collier. 2006. *The development of a schema for the annotation of terms in the BioCaster disease detection/tracking system*. Proc. of the Int'l Workshop on Biomedical Ontology in Action (KR-MED 2006), Baltimore, Maryland, USA, November 8, pp. 77-85, 2006.

A. Kawazoe, H. Chanlekha, M. Shigematsu and N. Collier. 2007. *Structuring an event ontology for disease outbreak detection*. The 2nd International Symposium on Languages in Biology and Medicine (LBM) (accepted to appear).

MedISys. 2007. Medical Intelligence System. http://medusa.jrc.it/medisys.

S. Morse S. 2007. *Global Infectious Disease Surveillance And Health Intelligence*. Health Affairs, 26(4):1069-1077, 2007.

ProMED-mail. 2001. The Program for Monitoring Emerging Diseases. http://www.promedmail.org.

A Mechanism to Provide Language-Encoding Support and an NLP Friendly Editor

Anil Kumar Singh
Language Technologies Research Centre
IIIT, Hyderabad, India
anil@research.iiit.ac.in

Abstract

Many languages of the world (some with very large numbers of native speakers) are not yet supported on computers. In this paper we first present a simple method to provide an extra layer of easily customizable language-encoding support for less computerized languages. We then describe an editor called Sanchay Editor, which uses this method and also has many other facilities useful for those using less computerized languages for simple text editing or for Natural Language Processing purposes, especially for annotation.

1 Introduction

A large number of languages of the world are still not supported on computers. Some of them are spoken by a tens or hundreds of millions of people, so they will probably be supported in the near future. However, many languages may not be, simply because the number of people using them on computers, for whatever reason, is not large. Those who want to use these languages on computers, including the researchers working on those languages, will need support for these languages. A related problem is that of support for encodings, as many of these less computerized languages do not have one standard encoding that is used by all. Therefore, there is a need of a simple and easily customizable method of adding support for a new language or encoding. Such a method should require minimum technical knowledge from the user. In this paper, we will present a method of providing language and encoding support for less computerized languages.

Another need which we address in this paper is of an editor that not only makes use of the above mentioned method of language and encoding support, but also has many facilities useful for Natural Language Processing (NLP) researchers and linguists.

2 Language-Encoding Support

There is no exhaustive, commonly agreed upon list of encodings for many languages. Even the list of languages is not without dispute (e.g., whether Bhojpuri is a language or not). This implies that the conventional deterministic approach to language-encoding support based on the assumption that the possible languages and encodings are known in advance is not enough if we do not want to prevent the possibility of using any language or encoding used by a significant number of people, or even a rarely used endangered language.

Even though with the increasing use of Unicode based encodings, the problem has reduced for many languages, we still require a facility that can allow convenient use of new languages which are not covered in Unicode.

Therefore, what we need is a more customizable language-encoding support where it is very easy for the user or the developer to add support for some language-encoding. For this purpose, as many of the tasks should be automated as possible. This can be done by using NLP techniques. Even though many of the encodings used for less computerized languages are based on just font mappings, i.e., supporting them basically means providing an appropri-

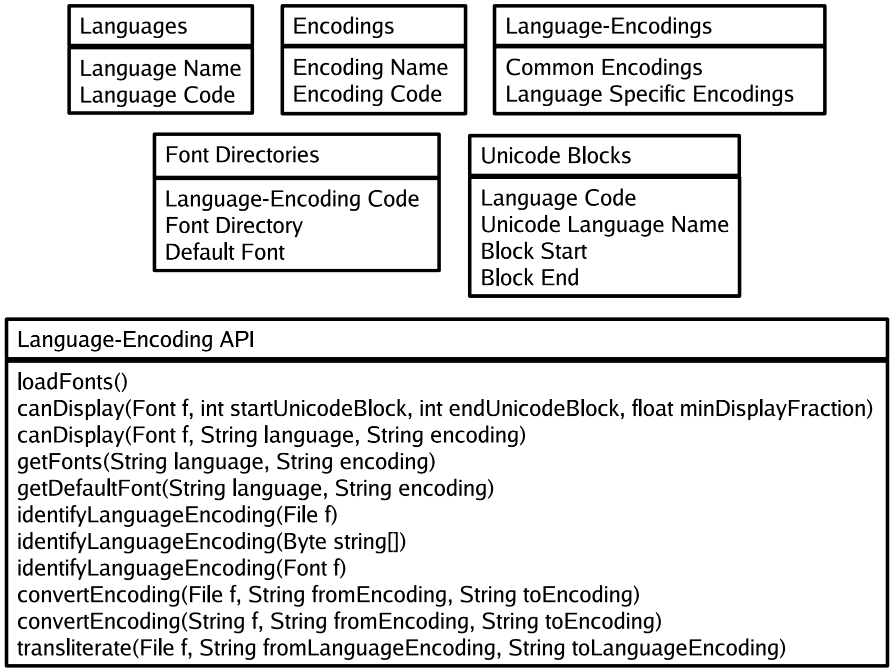


Figure 1: A customizable design for language-encoding support

ate font. This seems to be very simple, but the problem is that the user may not know which font to use. Moreover, providing basic support so that you can type once you have selected the font is not enough. The user might not even know what encoding some existing text is in. Then, the user might want to save the text in some other encoding. To provide user friendly support for language-encodings in a situation like this requires a more intelligent design.

Figure-1 shows a design for language-encoding support which addresses these problems. The main elements of this design are:

- Properties files listing languages, encodings, fonts, and their connections
- Language-encoding identification for text
- Language-encoding identification for fonts
- A language-encoding API
- Encoding converters

Currently, 15 languages and 10 encoding are supported. These are mostly all South Asian languages, apart from English, since the users till now were mostly from South Asia. A large number of freely available fonts have also been included in the distribution, but the user would probably like to add more fonts, which can be done easily just by adding the paths of the new fonts in a properties file. There is no need to install these fonts, irrespective of the operating systems. Also, more languages and encodings can be added quite easily. In most cases, to add a new language-encoding, the user just has to follow these steps:

1. Make a new entry in the properties files for each of these three: languages, encodings and language-encodings.
2. Specify the paths of all the fonts for that language-encoding in the properties file for fonts. These fonts just have to be on the system and their paths have to specified in the properties file. However, it may be preferable (for convenience) that they be stored in `fonts` directory of Sanchay.
3. Specify the default font in the properties file for default fonts.
4. If the new language uses a Unicode encoding, make an entry for the Unicode block corre-

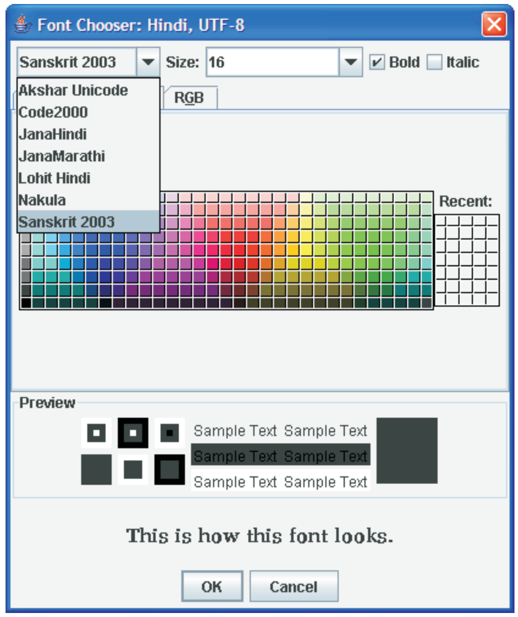


Figure 2: A font chooser listing fonts relevant to a specific language encoding pair

sponding to that language. This is not compulsory, but it will allow language specific listing of fonts for language-encoding pairs involving Unicode encodings.

In future, we will make this even more easy by providing a graphic user interface based wizard to go through these steps.

The editor can also use any input methods available from the operating system. New input methods can also be added as Java libraries. Such external Java libraries have just to be copied to the `ext-lib` directory of Sanchay. It is also possible to easily switch among input methods (Figure-4), whether provided by the operating system or included into (or add to) Sanchay. So, it is possible to enter text in multiple languages.

Note that, right now, this support for language-encodings is in the form of an extra platform independent layer on top of the support provided by operating systems. Such a layer could possibly be integrated into operating systems in future. This might, of course, require porting of the code for different operating systems and can be in-built into the operating system.

2.1 A More Intelligent Listing of Fonts

In the design used on all operating systems so far, when you want to view the list of fonts, what you get is a list of **all** the fonts installed on the system or at least all the fonts found by the operating system or the user program. This is not very user friendly for less computerized languages, because most of the fonts listed may not be meant for the language-encoding the user is interested in. What the user needs is the list of fonts relevant to the specific language-encoding she is interested in. In our design, this is what the user will see (Figure-2), when the user views the list of fonts. Of course, we can also give the user the option to see all the fonts installed on the system.

2.2 Language-Encoding Identification

Another important element of the design is a language-encoding identification tool that is integrated into the language-encoding support module so that if the user opens a file and does not know the language or encoding of the text, the tool can automatically identify the language-encoding of the text. The language-encoding identification tool is based on byte based n-gram models using a distributional similarity measures (Singh, 2006a). This tools is computationally quite a light one as the amount of

data required for training is very small and it has been found to be one of the most accurate language-encoding systems currently available. The user can make it even faster by removing those language-encodings which she may not be interested in. This will require only a change in the relevant properties file.

2.3 Encoding Conversion

There is also a wrapper module for calling any installed or built in encoding converter for languages which use more than one encodings. The user can easily convert the encoding of the text depending on her needs and the availability of a relevant encoding converter. It is also possible to easily add new encoding converters.

3 Sanchay Editor

Although numerous text editors, even free and open source ones, are available, the simple open source editor that we are going to describe in this section (Figure-3) is based on the language-encoding support mentioned earlier and is also closely integrated with Sanchay[1], a collection of tools and APIs for NLP. The editor is implemented as a customizable GUI component that can be easily included in any Java application. The notable features of this editor are:

- Uses customizable language-encoding support as described earlier.
- Can automatically identify language-encoding of the text using a byte based *n*-gram modeling (Singh, 2006a).
- The font chooser (Figure-2) shows only the fonts applicable for the language-encoding.
- Text can be preprocessed for NLP or annotation purposes from this editor.
- The formats used for annotation can be detected and validated from the editor.
- Specialized annotation interfaces can be launched to edit the annotated files (in text format) opened in this editor.
- Since the editor is implemented in Java, it can be used on any platform on which Java (JRE or JDK version 1.5 or higher) is installed.

[1] http://ltrc.iiit.ac.in/anil/Sanchay-EILMT and http://sourceforge.net/projects/nlp-sanchay

Some of the facilities are described in the following sub-sections.

3.1 Validation of Annotation Formats

If the user is directly editing a document which is annotated with POS tags, chunks or is syntactically annotated, it is possible to automatically validate the annotation format of the document. A text box below the main editing panel shows the errors in format, if any. Usually, annotation is performed by using some annotation interface, but since the annotated data is stored as simple text, the document can be edited or annotated directly from a text editor. The format validation facility has been included to ensure that after any such editing or annotation, the document is still in the correct format, as it is easy for users to make format related mistakes.

3.2 Format Conversion

Sanchay annotation interfaces allow annotation at various levels like POS tagging, chunking, syntactic (treebank) annotation etc. Currently four different formats are recognized by the system: raw text without annotation, POS tagged format where each sentence is simply a sequence of word and POS tag pairs separated by some symbol like underscore, 'bracket form' which allows POS tagged and chunked data to be represented (including recursion), and Shakti Standard Format (SSF)[2]. The editor allows the user to convert the data from one format to another.

3.3 Document Statistics

The user can also get a statistics about the document, such as the number of words, the number of sentences, the number of characters, and their respective frequencies etc. These statistics are according to the format of the document, i.e., if the document is in SSF format, then the document will be parsed and the statistics will be about the annotated document and the elements of the format, e.g. <Sentence> tag will not be counted: only actual words (or POS tags etc.) in the annotated document will be counted. Such statistics can also be obtained for a number of documents, i.e., a corpus, not just the current document. This can be a very useful facility for working on annotated corpus.

[2] www.elda.org/en/proj/scalla/SCALLA2004/sangalsharma.pdf

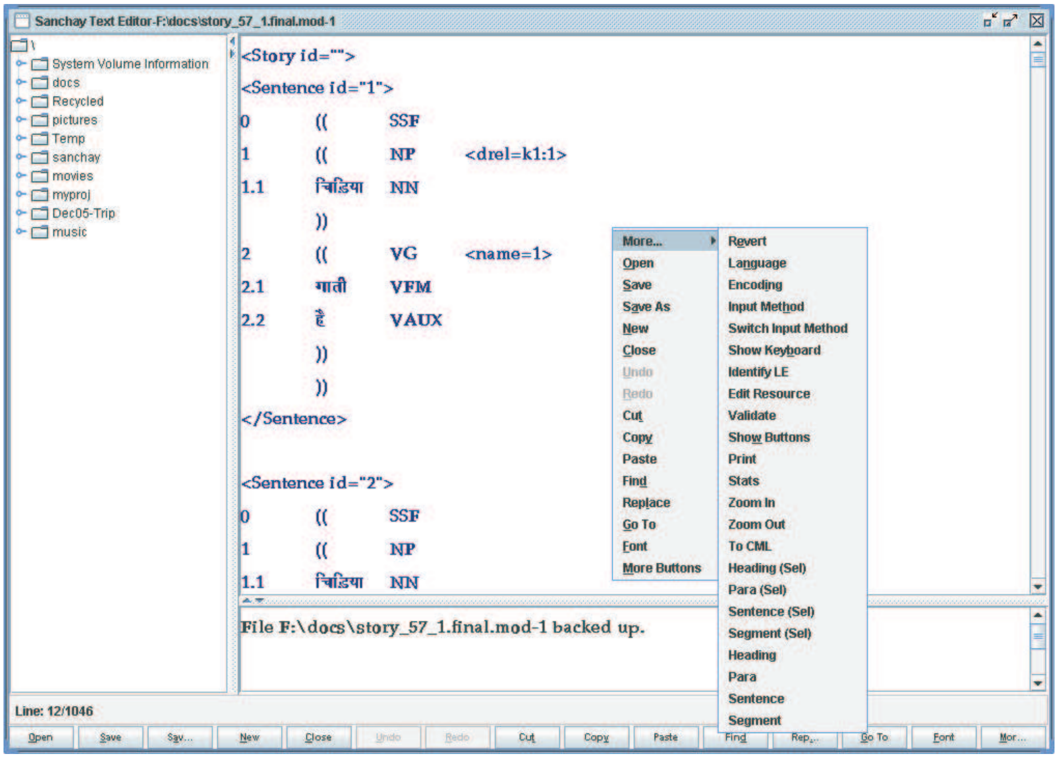


Figure 3: A multipurpose editor for NLP for South Asian languages

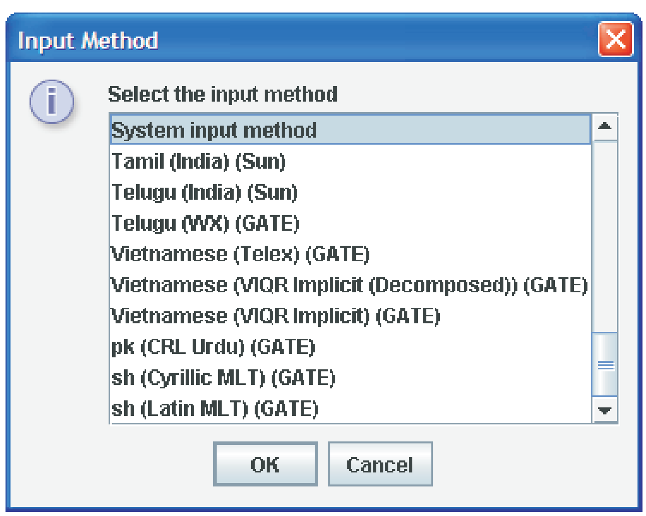


Figure 4: Input methods currently supported

3.4 Integration with Annotations Interfaces

The editor is built into Sanchay in such a way that it is possible to open different views of a document, depending on the annotation format. For example, if the currently opened document is in SSF format, then the same document can be opened in the Sanchay Syntactic Annotation Interface just by clicking on a button or a context menu item. The opposite is also possible, i.e., if a document is open in the Syntactic Annotation Interface, then it can be directly opened into the Sanchay Editor as a simple text file.

3.5 Some Other Facilities

Apart from the above mentioned facilities, Sanchay Editor also has the usual facilities available in text editors such as find and replace (with regular expressions and also in the batch mode), reverting to the saved version, automatic periodic backup etc.

4 Facilities Being Integrated

Some other facilities that have already been implemented and are going to be integrated into the Sanchay Editor include a better spell checker for South Asian languages based on a Computational Phonetic Model of Scripts or CPMS (Singh, 2006b). This model provides a method to calculate the phonetic and orthographic similarity (*surface similarity*) of words or strings. Another facility is the identification of languages and encoding in a multilingual document (Singh and Gorla, 2007a). This is an extension of the language-encoding identification tools described earlier and is the first systematic work on the problem of identification of languages and encoding in a multilingual document. When this tool is integrated into the editor, the user will be able to open a multilingual document and the system will automatically identify the sections in different languages and display them accordingly, even if the document has not been encoded using Unicode. Of course, identification is not 100% accurate at present, but we are working on improving it. Another already implemented facility that is going to be added is fuzzy text search (Singh et al., 2007c). It is also mainly based on the idea of calculating surface similarity using the CPMS. Fuzzy text search based on this method performs better than the traditional methods. Yet another facility to be added is a more discerning mechanism for transliteration (Surana and Singh, 2008). The first important idea in this mechanism is to use different methods for transliteration based on the word origin (identified using a modified version of the language-encoding tool). The second major idea is to use fuzzy text matching for selecting the best match. This method also has outperformed other methods.

There is a plan to extend the editor to allow direct annotation. We will begin by providing support for discourse annotation and other similar annotations.

5 Conclusions

In this paper we presented a simple but effective method of providing an easily customizable extra layer of language-encoding support for less computerized languages. We also described Sanchay Editor, which uses this method of language-encoding support and has many other facilities that may be useful for NLP researchers as well as those who just need a simple text editor for language-encodings not usually supported on computers. Sanchay Editor is closely integrated with a collection of NLP tools and APIs called Sanchay.

References

Anil Kumar Singh and Jagadeesh Gorla. 2007a. Identification of languages and encodings in a multilingual document. In *Proceedings of the 3rd ACL SIGWAC Workshop on Web As Corpus*, Louvain-la-Neuve, Belgium.

Anil Kumar Singh, Harshit Surana, and Karthik Gali. 2007c. More accurate fuzzy text search for languages using abugida scripts. In *Proceedings of ACM SIGIR Workshop on Improving Web Retrieval for Non-English Queries*, Amsterdam, Netherlands.

Anil Kumar Singh. 2006a. Study of some distance measures for language and encoding identification. In *Proceedings of ACL 2006 Workshop on Linguistic Distance*, Sydney, Australia.

Anil Kumar Singh. 2006b. A computational phonetic model for indian language scripts. In *Constraints on Spelling Changes: Fifth International Workshop on Writing Systems*, Nijmegen, The Netherlands.

Harshit Surana and Anil Kumar Singh. 2008. A more discerning and adaptable multilingual transliteration mechanism for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing (To appear)*, Hyderabad, India.

NLP Applications of Sinhala: TTS & OCR

Ruvan Weerasinghe, Asanka Wasala, Dulip Herath and Viraj Welgama
Language Technology Research Laboratory,
University of Colombo School of Computing,
35, Reid Avenue, Colombo 00700, Sri Lanka
{arw,raw,dlh,wvw}@ucsc.cmb.ac.lk

Abstract

This paper brings together the practical applications and the evaluation of the first Text-to-Speech (TTS) system for Sinhala using the Festival framework and an Optical Character Recognition system for Sinhala.

1 Introduction

Language Technology Research Laboratory [†] (LTRL) of the University of Colombo School of Computing (UCSC), was established in 2004 evolving from work engaged in by academics of the university since the early 1990's in local language computing in Sri Lanka.

Under the scope of the laboratory, numerous Natural Language Processing projects are being carried out with the relevant national bodies, international technology partners, local industry and the wider regional collaboration particularly within the PAN Localization Initiative[*]. The Sri Lankan component of the PAN Localization Project concentrated on developing some of the fundamental resources needed for language processing and some software tools for immediate deployment at the end of the project. Among the resources produced is a Sinhala Language Corpus of 10m words, and a tri-lingual Sinhala-English-Tamil lexicon. The two main software tools developed include a Sinhala Text-to-Speech (TTS) system and an Optical Character Recognition (OCR) system for recognizing commonly used Sinhala publications.

This paper focuses primarily on the end-user applications developed under the above project; Sinhala TTS system and OCR system. The paper describes the practical applications of these tools and evaluates it in the light of experience gained so far.

The rest of this paper is organized as follows: Section 2 gives an overview of the Sinhala TTS system; Section 3 describes the Sinhala OCR system. A summary along with future research directions and improvements are discussed in the last section.

2 Sinhala Text-to-Speech System

Sighted computer users spend a lot of time reading items on-screen to do their regular tasks such as checking email, fill out spreadsheets, gather information from internet, prepare and edit documents, and much more. However visually impaired people cannot perform these tasks without an assistance from other, or without using assistive technologies.

A TTS (text-to-speech) system takes computer text and converts the words into audible speech (Dutoit, 1997). With a TTS engine, application, and basic computer hardware, one can listen to computer text instead of reading it. A Screen Reader (2007) is a piece of software that attempts to identify and read-aloud what is being displayed on the screen. The screen reader reads aloud text within a document, and it also reads aloud information within dialog boxes and error messages. In other words, the primary function of any-screen reading system is to become the "eye" of the visually impaired computer user. These technologies enable blind or visually impaired people to do things that they could not perform before by them-

[†] See website: http://www.ucsc.cmb.ac.lk/ltrl
[*] See project website: http://www.panl10n.net

selves. As such, text-to-speech synthesizers make information accessible to the print disabled.

Within Sri Lanka, there is a great demand for a TTS system in local languages, particularly a screen reader or web browser for visually impaired people. In the case of the Tamil language, work done in India could be used directly. Until the LTRL of UCSC initiatives were launched in 2004, there was no viable TTS system found developed for Sinhala, the mother tongue of 74 % Sri Lankans (Karunatillake, 2004).

A project was launched to develop a 'commercial grade' Sinhala text-to-speech system in UCSC in year 2004. Later, it was extended to develop a Screen Reader which can be used by visually impaired persons for reading Sinhala texts.

The Sinhala TTS system was implemented based on the Festival speech synthesizer (Taylor et al., 1998). The Festival speech synthesis system is an open-source, stable and portable multilingual speech synthesis system developed at Center for Speech Technology Research (CSTR), University of Edinburgh (Taylor et al., 1998, Black and Lenzo, 2003). TTS systems have been developed using the Festival framework for different languages, including English, Japanese, Welsh, Turkish, Hindi, and Telugu (Black and Lenzo, 2003). However, efforts are still continuing to develop a standard Sinhala speech synthesizer in Sri Lanka.

The Sinhala text-to-speech system is developed based on the diphone concatenation approach. Construction of a diphone database and implementation of the natural language processing modules were key research areas explored in this project. In this exercise, 1413 diphones were determined. The diphones were extracted from nonsensical words, and recordings were carried out in a professional studio. Moreover, language specific scripts (phone, lexicon, tokenization) and speaker specific scripts (duration and intonation) were defined for Sinhala. It is worthy to mention the development of context-sensitive letter-to-sound conversion rule set for Sinhala. Incorporation of a high accuracy native syllabification routine (Weerasinghe et al., 2005) and implementation of comprehensive text analysis facilities (capable of producing the accurate pronunciation of the elements such as numbers, currency symbols, ratios, percentages, abbreviations, Roman numerals, time expressions, number ranges, telephone numbers, email addresses, English letters and various other symbols) have been found unique for the language (Weerasinghe et al., 2007). Despite the Festival's incomplete support for UTF-8, the above rules were rewrittenin UTF-8 multi-byte format following the work done for Telugu language (Kamisetty, 2006).

The current Sinhala TTS engine accepts Sinhala Unicode text and converts it into Speech. A male voice has been incorporated. Moreover, the system has been engineered to be used in deferent platforms, operating systems (i.e. Linux and Windows) and by different software applications (Weerasinghe et al., 2007).

2.1 Applications of TTS Synthesis Engine

Sinhala text is made accessible via two interfaces, by the TTS engine. A standalone software named *"Katha Baha"* primarily reads documents in Sinhala Unicode text format aloud. The same application can also be used to record the synthesized speech.

In this way, local language news papers and text books can be easily transformed into audio materials such as CDs. This software provides a convenient way to disseminate up-to-date news and information for the print disabled. e.g. Newspaper company may podcast their news paper, enabling access for print disabled and everyone else. Furthermore, the same application can be utilized to produce Sinhala digital talking books. To ensure the easy access by print disabled, keyboard short cuts are provided.

Owing to the prevalent use of Windows among the visually impaired community in Sri Lanka, it becomes essential that a system is developed within the Windows environment which offers Sinhala speech synthesis to existing applications. The standard speech synthesis and recognition interface in Microsoft Windows is the Microsoft Speech Application Programming Interface (MS-SAPI) (Microsoft Corporation, n.d.). MS-SAPI enabled applications can make use of any MS-SAPI enabled voice that has been installed in Windows. Therefore, steps were taken to integrate Sinhala voice into MS-SAPI. As a result, the MS-SAPI compliant Sinhala voice is accessible via any speech enabled Windows application. The Sinhala voice is proved to work well with "Thunder"[‡] a freely available screen reader for Windows. Additionally, steps were taken to translate and integrate

[‡] Available from: http://www.screenreader.net/

common words found related to Thunder screen reader (e.g. link="සබැඳිය", list item= "ලැයිස්තු අයිතම") (Weerasinghe et al., 2007).

Since most Linux distributions now come with Festival pre-installed, the integration of Sinhala voice in such platforms is very convenient. Furthermore, the Sinhala voice developed here was made accessible to GNOME-Orca and Gnopernicus - powerful assistive screen reader software for people with visual impairments.

It is noteworthy to mention that for the first time in Sri Lankan history, the print disabled community will be able to use computers in their local languages by using the current Sinhala text-to-speech system.

2.2 Evaluation of the Text-to-Speech Synthesis Engine

Text-to-speech systems have been compared and evaluated with respect to intelligibility (understandability of speech), naturalness, and suitability for used application (Lemmetty, 1999). As the Sinhala TTS system is a general-purpose synthesizer, a decision was made to evaluate it under the intelligibility criterion. Specially, the TTS system is intended to be used with screen reader software by visually impaired people. Therefore, intelligibility is a more important feature than the naturalness.

A Modified Rhyme Test (MRT) (Lemmetty, 1999), was designed to test the Sinhala TTS system. The test consists of 50 sets of 6 one or two syllable words which makes a total set of 300 words. The words are chosen to evaluate phonetic characteristics such as voicing, nasality, sibilation, and consonant germination. Out of 50 sets, 20 sets were selected for each listener. The set of 6 words is played one at the time and the listener marks the synthesized word. The overall intelligibility of the system measured from 20 listeners was found to be 71.5% (Weerasinghe et al., 2007).

3 Optical Character Recognition System

Optical Character Recognition (OCR) technology is used to convert information available in the printed form into machine editable electronic text form through a process of image capture, processing and recognition (Optical Character Recognition, 2007).

There are three essential elements to OCR technology. Scanning – acquisition of printed documents as optical images using a device such as flatbed scanner. Recognition- involves converting these images to character streams representing letters of recognized words and the final element involves accessing or storing the converted text.

Many OCR systems have been developed for recognizing Latin characters (Weerasinghe et al., 2006). Some OCR systems have been reported to have a very high accuracy and most of such systems are commercial products. Leaving a land mark, a Sinhala OCR system has been developed at UCSC (Weerasinghe et al., 2006).

Artificial Neural Network (ANN) and Template Matching are two popular and widely used algorithms for optical character recognition. However, the application of above algorithms to a highly inflected languages such as Sinhala is arduous due to the high number of input classes. Empirical estimation of least number of input classes needed for training a neural net for Sinhala character recognition suggested about 400 classes (Weerasinghe et al., 2006). Therefore, less-complicated K-nearest neighbor algorithm (KNN) was employed for the purpose of Sinhala character recognition.

The current OCR system is the first ever reported OCR system for Sinhala and is capable of recognizing printed Sinhala letters typed using widely used fonts in the publishing industry. The recognized content is presented as editable Sinhala Unicode text file (Weerasinghe et al., 2006).

A large volume of information is available in the printed form. The current OCR system will expedite the process of digitizing this information. Moreover, the information available via printed medium is inaccessible to the print disabled, and the OCR system, especially when coupled with Sinhala TTS, will provide access to these information for the print disabled.

3.1 Evaluation of the Optical Character Recognition System

The performance of the Sinhala OCR system has been evaluated using 18000 sample characters for Sinhala. These characters have been extracted from various books and newspapers (Weerasinghe et al., 2006). Performance of the system has been evaluated with respect to different best supportive fonts. The results have been summarized in the Table 1 (Weerasinghe et al., 2006).

Font	*FM*	*DL*	*Lakbima*	*Letter*
% Recog.	97.17	96.26	89.89	95.81

Table 1. Experimental Results of Classification*

From this evaluation it can be concluded that the current Sinhala OCR has average accuracy of 95% (Weerasinghe et al., 2006).

4 Conclusion and Future Work

This paper brings together the development of a diphone voice for Sinhala based on the Festival speech synthesis system and an Optical Character Recognizer for Sinhala.

Future work on the Sinhala TTS engine will mainly focus on improving the prosody modules. A speech corpus containing 2 hours of speech has been already recorded. The material is currently being segmented, and labeled. We are also planning to improve the duration model using the data obtained from the annotated speech corpus. It is also expected to develop a female voice in near future. The current Sinhala OCR system is font dependent. Work is in progress to make the OCR system font independent and to improve the accuracy. Sinhala OCR and the TTS systems, which are currently two separate applications, will be integrated enabling the user friendliness to the print disabled.

A number of other ongoing projects are aimed at developing resources and tools such as a POS tag set, a POS tagger and a tagged corpus for Sinhala, an on-the-fly web page translator, a translation memory application and several language teaching -learning resources for Sinhala, Tamil and English.

All resources developed under this project are made available (under GNU General Public License) through the LTRL website.

Acknowledgement

This work was made possible through the PAN Localization Project, (http://www.PANL10n.net) a grant from the International Development Research Center (IDRC), Ottawa, Canada, administered through the Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

* FM – "FM Abhaya", DL – "DL Manel Bold", Letter – "Letter Press"

References

Alan W. Black and Kevin A. Lenzo. 2003. *Building Synthetic Voices*, Language Technologies Institute, Carnegie Mellon University and Cepstral LLC. Retrieved from http://festvox.org/bsv/.

Microsoft Corporation. (n.d.). *Microsoft Speech SDK Version 5.1*. Retrieved from:

http://msdn2.microsoft.com/en-/library/ms990097.aspx

T. Dutoit. 1997. *An Introduction to Text-to-Speech Synthesis*, Kluwer Academic Publishers, Dordrecht, Netherlands.

C. Kamisetty, S.M. Adapa. 2006. *Telugu Festival Text-to-Speech System*. Retrieved from:

http://festival-te.sourceforge.net/wiki/Main_Page

W.S. Karunatillake. 2004. *An Introduction to Spoken Sinhala, 3rd edn.*, M.D. Gunasena & Co. ltd., 217, Olcott Mawatha, Colombo 11.

Sami Lemmetty. 1999. *Review of Speech Synthesis Technology*, MSc. thesis, Helsinki University of Technology.

Screen Reader. 2007. *Screen Reader*. Retrieved from: http://en.wikipedia.org/wiki/Screen_reader.

Optical Character Recognition. 2007. *Optical Character Recognition.* Retrieved from: http://en.wikipedia.org/wiki/Optical_character_recognition

P.A Taylor, A.W. Black, R.J. Caley. 1998. The Architecture of the Festival Speech Synthesis System, *Third ESCA Workshop in Speech Synthesis*, Jenolan Caves, Australia. 147-151.

Ruvan Weerasinghe, Asanka Wasala, Kumudu Gamage. 2005. A Rule Based Syllabification Algorithm for Sinhala, *Proceedings of 2nd International Joint Conference on Natural Language Processing (IJCNLP-05).* Jeju Island, Korea. 438-449.

Ruvan Weerasinghe, Dulip Lakmal Herath, N.P.K. Medagoda. 2006. A KNN based Algorithm for Printed Sinhala Character Recognition, *Proceedings of 8th International Information Technology Conference*, Colombo, Sri Lanka

Ruvan Weerasinghe, Asanka Wasala, Viraj Welgama and Kumudu Gamage. 2007. Festival-si: A Sinhala Text-to-Speech System, *Proceedings of 10th International Conference on Text, Speech and Dialogue (TSD 2007)*, Pilseň, Czech Republic, September 3-7, 2007. 472-479

POLLy: A Conversational System that uses a Shared Representation to Generate Action and Social Language

Swati Gupta
Department of Computer Science, Regent Court
211 Portobello Street
University of Sheffield
Sheffield, UK
s.gupta@dcs.shef.ac.uk

Marilyn A. Walker
Department of Computer Science, Regent Court
211 Portobello Street
University of Sheffield
Sheffield, UK
m.walker@dcs.shef.ac.uk

Daniela M. Romano
Department of Computer Science, Regent Court
211 Portobello Street
University of Sheffield
Sheffield, UK
d.romano@dcs.shef.ac.uk

Abstract

We present a demo of our conversational system POLLy (POliteness in Language Learning) which uses a common planning representation to generate actions to be performed by embodied agents in a virtual environment and to generate spoken utterances for dialogues about the steps involved in completing the task. In order to generate socially appropriate dialogue, Brown and Levinson's theory of politeness is used to constrain the dialogue generation process.

1 Introduction

Research in Embodied Conversational Agents (ECAs) has explored embedding ECAs in domain-specific Virtual Environments (VE) where users interact with them using different modalities, including Spoken Language. However, in order to support dialogic interaction in such environments, an important technical challenge is the synchronization of the ECA Spoken Interaction module with the ECA non-verbal actions in the VE. We propose an approach that uses a common high level representation which is broken down to simpler levels to generate the agents' verbal interaction and the agents' non-verbal actions synchronously for task-oriented applications that involve performing some actions to achieve a goal, while talking about the actions using natural language.

In previous work, Bersot et al (1998) present a conversational agent called Ulysses embedded in a collaborative VE which accepts spoken input from the user and enables him or her to navigate within the VE. They use a 'reference resolver' which maps the entities mentioned in utterances to geometric objects in the VE and to actions.

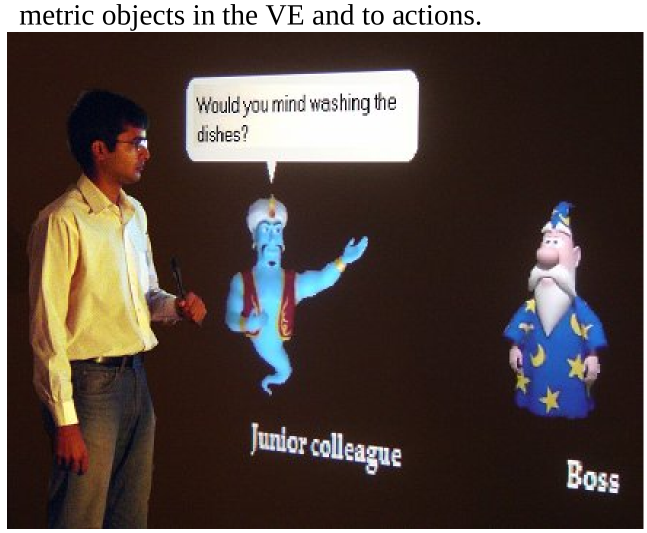


Figure 1. A user interacting with the Agents

Max, a VR based conversational agent by Kopp et al (2003) allows multimodal conversational abilities for task-oriented dialogues in virtual construction tasks. It builds on a database of utterance templates which contain the verbal part, augmented with accompanying gestures and their cross-modal affiliation. In order to deal with the vagueness of language in specifying spatial relations in virtual space, the K_2 system (Takenobu et al 2003) proposed a bilateral symbolic and numeric representation of locations, to bridge the gap between language processing (a symbolic system), and animation generation (a continuous system). K_2 extracts a user's goal from the utterance and

translates it into animation data. The FearNot! demonstrator by Paiva et al (2005) provides training to kids against bullying via virtual drama in which one virtual character plays the role of a bully and the other plays the role of victim, who asks the child for advice. FearNot!'s spoken interaction is template-based where the incoming text from the child is matched against a set of language templates. The information about the character's action is defined in a collection which contains the utterance to be spoken as well as the animation. Eichner et al (2007) describe an application in which life-like characters present MP3 players in a virtual showroom. An XML scripting language is used to define the content of the presentation as well as the animations of the agents. A more expressive agent, Greta, developed by Pelachaud et al (Poggi et al, 2005) is capable of producing socially appropriate gestures and facial expressions, and used is in an evaluation of gesture and politeness as reported in Rehm and André (2007).

Since these ECAs function in scenarios where they interact with the world, other agents, and the user, they must be 'socially intelligent' (Dautenhahn, 2000) and exhibit social skills. Our work is based on the hypothesis that the relevant social skills include the ability to communicate appropriately, according to the social situation, by building on theories about the norms of human social behaviour. We believe that an integral part of such skills is the correct use of politeness (Brown & Levinson, 1987; Walker et al 1997). For instance, note the difference in the effect of requesting the hearer to clean the floor by saying 'You must clean the spill on the floor now!' and 'I know I'm asking you for a big favour but could you kindly clean the spill on the floor?'

According to Brown and Levinson (1987) (henceforth B&L), choices of these different forms are driven by sociological norms among human speakers. Walker et al (1997) were the first to propose and implement B&L's theory in ECAs to provide interesting variations of character and personality in an interactive narrative application. Since then B&L's theory has been used in many conversational applications e.g. animated presentation teams (André et al 2000; Rehm & André, 2007), real estate sales (Cassell & Bickmore, 2003), and tutorials (Johnson et al, 2004; Johnson et al, 2005; Porayska-Pomsta 2003; Wang et al 2003). Rehm & André (2007) show that gestures are used consistently with verbal politeness strategies and specific gestures can be used to mitigate face threats. Work in literary analysis has also argued for the utility of B&L's theory, e.g. Culpeper (1996) argues that a notion of 'impoliteness' in dramatic narratives creates conflict by portraying verbal events that are inappropriate in real life. Thus impoliteness often serves as a key to move the plot forward in terms of its consequences.

This demo presents our Conversational System POLLy which produces utterances with a socially appropriate level of politeness as per the theory of Brown and Levinson. We have implemented POLLy in a VE for the domain of teaching English as a second language (ESL). It is rendered in our VE RAVE at Sheffield University as well as on a normal computer screen, as explained in section 3. Figure 1 shows a user interacting with POLLy in RAVE. Since RAVE is not portable, we will demonstrate POLLy on the computer screen where the user will be able to verbally communicate with the agents and the agents will respond with computationally generated utterances with an appropriate level of politeness as per a given situation.

2 POLLy's Architecture

POLLy uses a shared representation for generating actions to be performed by the ECAs in the virtual domain on one hand, and on the other, for generating dialogues to communicate about the actions to be performed. It consists of three components: A Virtual Environment (VE), a Spoken Language Generation (SLG) system and a Shared AI Planning Representation for VE and SLG as illustrated in Figure 2.

A classic STRIPS-style planner called GraphPlan (Blum & Furst, 1997) produces, given a goal e.g. cook pasta, a plan of the steps involved in doing so (Gupta et al., 2007). POLLy then allocates this plan to the Embodied Conversational Agents (ECA) in the VE as a shared collaborative plan to achieve the cooking task with goals to communicate about the plan via speech acts (SAs), needed to accomplish the plan collaboratively, such as Re quests, Offers, Informs, Acceptances and rejections (Grosz, 1990; Sidner, 1994; Walker, 1996). It also allocates this plan to the SLG which generates variations of the dialogue based on B&L's theory of politeness that realizes this collaborative plan as in (André et al,2000;Walker et al, 1997).

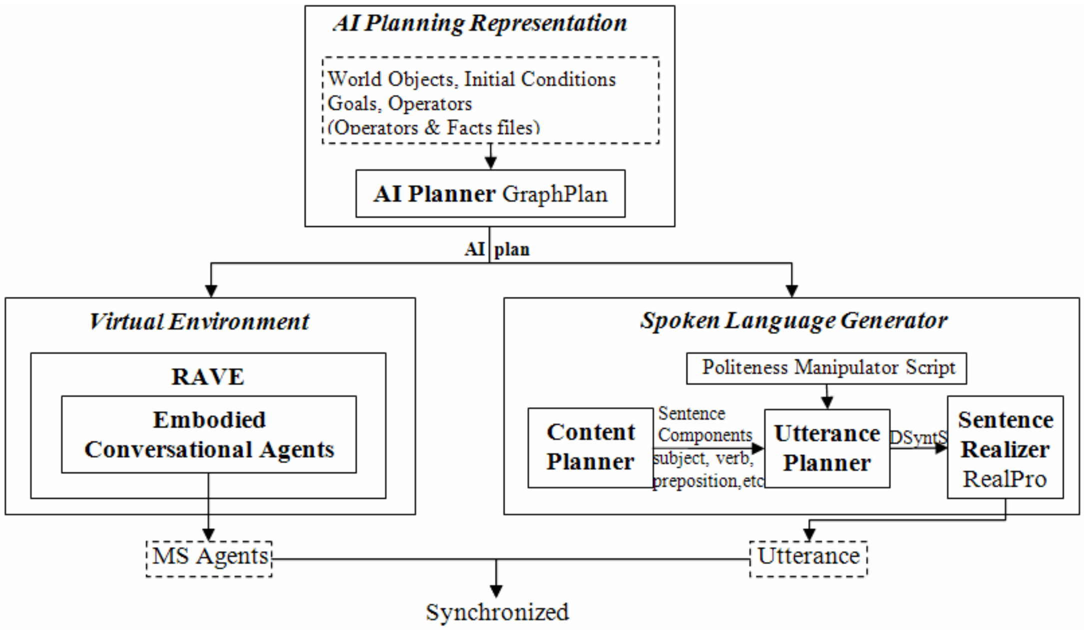


Figure 2: POLLY's Architecture

The SLG (Gupta et al., 2007) is based on a standard architecture (Dale & Reiter, 1995) with three components: Content planning, utterance planning and surface realization. See Figure 2. The politeness strategies are implemented through a combination of content selection and utterance planning. The linguistic realizer RealPro is used for realization of the resulting utterance plan (Lavoie & Rambow, 1997), which takes a dependency structure called the Deep-Syntactic Structure (DSyntS) as input and realizes it as a sentence string. The Content Planner interfaces to the AI Planner, selecting content from the preconditions, steps and effects of the plan. According to B&L, direct strategies are selected from the steps of the plan, while realizations of preconditions and negating the effects of actions are techniques for implementing indirect strategies. The content planner extracts the components of the utterances to be created, from the plan and assigns them their respective categories, for example, lexeme get/add under category verb, knife/oil under direct object etc and sends them as input to the Utterance Planner. The Utterance Planner then converts the utterance components to the lexemes of DSyntS nodes to create basic DsyntS for simple sentences, which are then transformed to create variations as per B&L's politeness strategies, with the 'politeness manipulator script'. For realizing these B&L strategies, transformations to add lexical items such as 'please', 'if you don't mind', and 'mate' were added to the DSyntS to make a sentence less or more polite.

Some example dialogues are shown in section 3. In the VE, the human English language learner is able to interact with the Embodied Conversational Agent and plays the part of one of the agents in order to practice politeness real-time.

2.1 Brown and Levinson's theory

B&L's theory states that speakers in conversation attempt to realize their speech acts (SAs) to avoid threats to one another's face, which consists of two components. Positive face is the desire that at least some of the speaker's and hearer's goals and desires are shared by other speakers. Negative face is the want of a person that his action be unimpeded by others. Utterances that threaten the conversants' face are called Face Threatening Acts (FTAs). B&L predict a universal of language usage that the choice of linguistic form can be determined by the predicted Threat θ as a sum of 3 variables: P: power that the hearer has over the speaker; D: social distance between speaker & hearer; and R: a ranking of imposition of the speech act. Linguistic strategy choice is made according to the value of the Threat θ. We follow Walker et al.'s (1997) four part classification of strategy choice.

The Direct strategy is used when θ is low and executes the SA in the most direct, clear and

unambiguous way. It is usually carried out either in urgent situations (Please Help!), or where the face threat is small as in "I have chopped the vegetables" or if the speaker has power over the hearer, "Did you finish your homework today?"

The Approval strategy (Positive Politeness) is used for the next level of threat θ - this strategy is oriented towards the need for the hearer to maintain a positive self-image. Positive politeness is primarily based on how the speaker approaches the hearer, by treating him as a friend, a person whose wants and personality traits are liked, for ex. by using friendly markers "Friend, would you close the door?"

The Autonomy Strategy (Negative Politeness) is used for high face threats, when the speaker may be imposing on the hearer, intruding on their space or violating their freedom of action. These face threats can be mitigated by using hedges, "I wonder if you would mind closing the door for me," or by minimizing imposition, "I just want to ask you if you could close the door."

The Indirect Strategy (Off Record) is the politest strategy and is used when θ is greatest. It usually has more than one attributable intention so that the speaker removes himself from any imposition. For ex. using metaphor and irony, rhetorical questions, understatement, hints etc. For instance, "Its cold in here," which implies a request to close the door.

2.2 Example Dialogues

Here are some example dialogues that illustrate the difference in the politeness strategies used in discourse contexts of varying power. Two Microsoft Agents, Merlin and Genie are involved in a conversation while cooking and cleaning in a kitchen together. Consider the difference in the degree of politeness in each situation.

Conversation 1: This is an example conversation in which Merlin is Genie's boss.

Agent (Speech act: Politeness strategy): Utterance

Merlin*: Would you mind washing the dishes? (Approval: RequestAct)*
Genie: Sure, I'll wash the dishes. (Direct:AcceptRequest)
Genie: I'm wondering whether I should boil the pasta.(Autonomy:Offer)
Merlin*: Yeah you can. (Direct: AcceptOffer)*
Merlin*: You've burnt the vegetables. (Direct: Inform)*
Genie: Yeah. (Direct: AcceptInform)
Genie: The oven is dirty. (Indirect: RequestAct)
Merlin*: I'm sorry I can't clean the oven. (Approval: RejectRequest)*
Genie: Ok. (Direct: AcceptReject)
Genie: If you don't mind, can I clean the burner? (Autonomy: Offer)
Merlin*: No, forget it. (Direct: RejectOffer)*
Merlin*: I must wipe the counter now. (Direct: Offer)*
Genie: Alright, if you insist. (Autonomy: AcceptOffer)
Genie: Do you know that you've broken the baking dish mate? (Approval: Inform)
Merlin*: Yes. (Direct: AcceptInform)*
Merlin*: I'll cook the vegetables. (Direct: Offer)*
Genie: No please don't bother yourself. (Autonomy: RejectOffer)
Genie: The vegetables are healthy. (Indirect: RequestAct)
Merlin*: Alright, I'll add the vegetables to the pasta. (Direct: AcceptRequest)*
Merlin*: Could you please add the cheese to the pasta for me? (Approval: RequestAct)*
Genie: Please don't mind but I can't do that. (Autonomy: RejectRequest)
Merlin*: Alright. (Direct: AcceptReject)*

Conversation 2: In this case Genie and Merlin are colleagues at an equal level in hierarchy.

Agent (Speech act: Politeness strategy): Utterance

Merlin*: Could you possibly clean the oven for me?(Approval:RequestAct)*
Genie: Sure. (Direct:AcceptRequest)
Genie: I'll clean the burner. (Direct:Offer)
Merlin*: Ok. (Direct:AcceptOffer)*
Merlin*: You've burnt the vegetables. (Direct:Inform)*
Genie: Yeah. (Direct:AcceptInform)
Genie: Would you mind washing the dishes? (Approval:RequestAct)
Merlin*: I'm sorry but I can't wash the dishes. (Approval:RejectRequest)*
Genie: Alright. (Direct:AcceptReject)
Genie: I must boil the pasta. (Direct:Offer)
Merlin*: No, thanks. (Direct:RejectOffer)*
Merlin*: I can wipe the counter. (Direct:Offer)*
Genie: Yeah you can. (Direct:AcceptOffer)
Genie: You've broken the baking dish. (Direct:Inform)
Merlin*: Yes. (Direct:AcceptInform)*
Merlin*: I'll cook the vegetables. (Direct:Offer)*
Genie: No, forget it. (Direct:RejectOffer)
Merlin*: Could you please add the vegetables to the pasta? (Approval:RequestAct)*
Genie: Please don't mind but I can't do that. (Approval:RejectRequest)
Merlin*: Ok. (Direct:AcceptReject)*
Genie: Will you please wipe the table mate? (Approval:RequestAct)
Merlin*: Sure. (Direct:AcceptRequest)*

3 Virtual Environment

We rendered POLLy with Microsoft Agent Characters (Microsoft, 1998) in our Virtual Environment RAVE at Sheffield University as well as on a desktop computer screen. RAVE consists of a 3-dimensional visualisation of computer-generated scenes onto a 10ft x 8ft screen and a complete 3D surround sound system driven by a dedicated computer. Since Microsoft Agents are 2D, they are not rendered 3D, but a life size image of the characters is visible to the users on the screen to make them appear believable. Figure 1 showed a user interacting with POLLy in RAVE. The MS Agent package provides libraries to program control using various developing environments like the .NET framework and visual studio and includes a voice recognizer and a text-to-speech engine. It also provides controls to embed predefined animations which make the characters' behaviour look more interesting and believable (Cassell & Thórisson, 1999). We have programmed MS agent in Visual C++ and have embedded these animations like gesturing in a direction, looking towards the other agents, blinking, tilting the head, extending arms to the side, raising eyebrows, looking up and down etc while the agents speak and listen to the utterances and holding the hand to the ear, extending the ear, turning the head left or right etc when the agents don't understand what the user says or the user doesn't speak anything.

The Agents share the AI plan to collaborate on it together to achieve the cooking task. Goals to communicate about the plan are also allocated to the agents as speech acts (SAs) such as Requests, Offers, Informs, Acceptances and Rejections, needed to accomplish the plan collaboratively. While interacting with the system using a high quality microphone, the user sees one or two agents on the screen and plays the part of the second or the third agent, as per the role given to him/her.

When we extend this to a real-time immersive Virtual Reality environment, a Virtual Kitchen in this case, the ECAs will actually perform the task of cooking a recipe together in the virtual kitchen while conversing about the steps involved in doing so, as laid out by the AI plan.

This setup makes it possible to design a 2x2x2 experiment to test three conditions: *Interactivity*, i.e. whether the user only sees the agents interacting on the screen vs. the user interacts with the agents by playing a role; *immersiveness of the environment*, i.e. rendering in RAVE vs. rendering on a desktop computer; and *culture*, i.e. the difference between the perception of politeness by people from different cultures as in (Gupta et al., 2007). We are now in the process of completing the design of this experiment and running it.

4 Conclusion

We presents a demo of our conversational system POLLy which implements MS Agent characters in a VE and uses an AI Planning based shared representation for generating actions to be performed by the agents and utterances to communicate about the steps involved in performing the action. The utterances generated by POLLy are socially appropriate in terms of their politeness level. The user will be given a role play situation and he/she will be able to have a conversation with the agents on a desktop computer, where some dialogic utterances would be allocated to the user. An evaluation of POLLy (Gupta et al, 2007; Gupta et al, 2008) showed that (1) politeness perceptions of POLLy's output are generally consistent with B&L's predictions for choice of form for discourse situation, i.e. utterances to strangers or a superior person need to be very polite, preferably autonomy oriented (2) our indirect strategies which should be the politest forms, are the rudest (3) English and Indian speakers of English have different perceptions of politeness (4) B&L implicitly state the equality of the P & D variables in their equation ($\theta = P + D + R$), whereas we observe that not only their weights are different as they appear to be subjectively determined, but they are also not independent.

References

André, E., Rist, T., Mulken, S.v., Klesen, M., & Baldes, S. 2000. *The automated design of believable dialogues for animated presentation teams.* In Embodied Conversational Agents (pp. 220–255). MIT Press.

Bersot, O., El-Guedj, P.O., God´ereaux, C. and Nugues. P. 1998. *A conversational agent to help navigation & collaboration in virtual worlds.* Virtual Reality,3(1):71–82.

Blum, A., Furst, M. 1997. *Fast Planning Through Planning Graph Analysis.* Artificial Intelligence 90.

Cassell, J. and Thórisson, K.R. 1999. *The Power of a Nod and a Glance: Envelope vs. Emotional Feedback*

in Animated Conversational Agents. Applied Artificial Intelligence 13: 519-538.

Cassell, J. and Bickmore, Timothy W. Negotiated Collusion. 2003. *Modeling Social Language and its Relationship Effects in Intelligent Agents.* User Model. User-Adapt.Interact. 13(1-2):89-132.

Culpeper, J. 1996. *(Im)politeness in dramatic dialogue.* Exploring the Language of Drama: From text to context. Routledge, London.

Dale, R. and Reiter, E. 1995. *Building Natural Language Generation Systems.* Studies in Natural Language Processing. Cambridge University Press.

Dautenhahn, K. 2000. *Socially Intelligent Agents: The Human in the Loop* (Papers from the 2000 AAAI Fall Symposium). The AAAI Press, Technical Report.

Eichner, T., Prendinger, H., André, E. and Ishizuka, M. 2007. *Attentive presentation agents.* Proc. 7th International Conference on Intelligent Virtual Agents (IVA-07), Springer LNCS 4722. pp 283-295.

Grosz, B.J., Sidner, C.L. 1990. *Plans for discourse.* In: Cohen, P.R., Morgan, J.L., Pollack, M.E. (eds.) Intentions in Communication, MIT Press, Cambridge.

Gupta, S., Walker, M.A., Romano, D.M. 2007. *How Rude are You?: Evaluating Politeness and Affect in Interaction.* Affective Computing & Intelligent Interaction (ACII-2007).

Gupta , S., Walker, M.A., Romano, D.M. 2008 (to be published). *Using a Shared Representation to Generate Action and Social Language for a Virtual Dialogue Environment.* AAAI Spring Symposium on Emotion, Personality and Social Behavior.

Johnson, L.W. and Rizzo, P. and Bosma, W.E. and Ghijsen, M. and van Welbergen, H. 2004. *Generating socially appropriate tutorial dialog.* In: ISCA Workshop on Affective Dialogue Systems. pp. 254-264.

Johnson, L., Mayer, R., André, E., & Rehm, M. 2005. *Cross-cultural evaluation of politeness in tactics for pedagogical agents.* Proc. of the 12th Int. Conf. on Artificial Intelligence in Education.

Kopp, S., Jung, B., Lessmann, N. and Wachsmuth, I. 2003. *Max – A multimodal assistant in virtual reality construction*. KI Zeitschift (German Magazine of Artificial Intelligence), Special Issue on Embodied Conversational Agents, vol.4, pp.11–17.

Lavoie, B., and Rambow, O. 1997. RealPro – a fast, portable sentence realizer. In Proc. Conference on Applied Natural Language Processing (ANLP'97).

Microsoft. 1998. *Developing for Microsoft Agent*. Microsoft Press.

op den Akker, H.J.A. and Nijholt, A. 2000. *Dialogues for Embodied Agents in Virtual Environments.* In: Natural Language Processing - NLP 2000, 2nd Int. Conf. pp. 358-369. LNAI 1835.

Paiva, A., Dias, J., & Aylett, R.S. 2005. *Learning by feeling: evoking empathy with synthetic characters.* Applied Artificial Intelligence: 19 (3-4), 235-266.

Poggi, I., Pelachaud, C., de Rosis, F., Carofiglio, V., De Carolis, B. 2005. *GRETA. A Believable Embodied Conversational Agent*. in O. Stock and M. Zancarano, eds, Multimodal Intelligent Information Presentation, Kluwer.

Prendinger, Helmut and Ishizuka, Mitsuru. 2001. *Let's talk! Socially intelligent agents for language conversation training.* IEEE Transactions on xSystems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 31, No. 5, pp 465-471.

Porayska-Pomsta, K. 2003. *Influence of Situational Context on Language Production: Modelling Teachers' Corrective Responses.* PhD Thesis. School of Informatics, University of Edinburgh.

Rehm, M. and André, E. 2007. *Informing the Design of Agents by Corpus Analysis.* Conversational Informatics, Edited by T. Nishida.

Sidner, C.L. 1994. *An artificial discourse language for collaborative negotiation.* In: Proc. 12th National Conf. on AI, pp. 814–819.

Takenobu, T., Tomofumi, K., Suguru, S., Manabu, O. 2003. *Bridging the Gap between Language and Action.* IVA 2003, LNAI 2792, pp. 127-135.

Traum, D., Rickel, J., Gratch, J., Marsella, S. 2003. *Negotiation over Tasks in Hybrid Human-Agent Teams for Simulation-Based Training.* Proceedings of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems.

Walker, M.A. 1996. *The effect of resource limits and task complexity on collaborative planning in dialogue.* Artificial Intelligence Journal 85, 1–2.

Walker, M., Cahn, J. and Whittaker, S. J. 1997. *Improving linguistic style: Social and affective bases for agent personality.* In Proc. Autonomous Agents'97. 96–105. ACM Press.

Wang, N., Johnson, W.L., Rizzo, P., Shaw,E., & Mayer, R. 2005. *Experimental evaluation of polite interaction tactics for pedagogical agents.* Proceedings of IUI '05. ACM Press.

Watts, Richard J. Ide, S. and Ehlich, K. 1992. Introduction, in Watts, R, Ide, S. and Ehlich, K. (eds.), *Politeness in Language: Studies in History, Theory and Practice.* Berlin: Mouton de Gruyter, pp.1-17.

Cross Lingual Information Access System for Indian Languages

CLIA Consortium
A Consortium of 11 Institutions as the Implementing Agency for the
Project "Development of Cross Lingual Information Access (CLIA) System" funded by Government of India, Ministry of Communications & Information Technology, Department of Information Technology (No. 14(5)/2006 – HCC (TDIL) Dated 29-08-2006)

1 Introduction

The **CLIA (Cross Lingual Information Access) Project** is a mission mode project funded by Government of India, Ministry of Communications & Information Technology, Department of Information Technology vide its approval No. 14(5)/2006 – HCC (TDIL), Dated 29-08-2006. It is being executed by a consortium of 11 academic and research institutions and industry partners, IIT Bombay, IIT Kharagpur, IIIT Hyderabad, AU-KBC Chennai, AU-CEG Chennai, ISI Kolkata, Jadavpur University Kolkata, C-DAC Pune, C-DAC Noida, Utkal University Bhubaneswar and STDC, DIT New Delhi. The final deliverables of the project at the end of two years will be a portal where:

- A user will be able to give a query in one Indian language and

- S/he will be able to access documents available in
  (a). the language of the query,
  (b). Hindi (if the query language is not Hindi), and
  (c). English

- Results will be presented to the user in the language of the query. The results can also be presented in the language in which the information originally resided. The languages involved are *Bengali, Hindi, Marathi, Punjabi, Tamil* and *Telugu.*

2 Motivation

With the tremendous growth of digital and online information repositories new opportunities and new problems are created for achieving information retrieval across different languages. Online documents are available internationally in many different languages. Cross Lingual Information Access (CLIA) systems makes it possible for users to directly access sources of information which may be available in languages other than the language of query. However in conventional information retrieval systems the user must enter a search query in the language of the documents in order to retrieve it. This requires that the user can formulate his/her queries in all possible languages and can decipher documents returned by the retrieval process. This restriction clearly limits the amount and type of information, which an individual user really has access to.

Cross-language information retrieval enables users to enter queries in languages they are familiar to, and uses language translation methods to retrieve documents originally created in other languages. Cross-Language Information Access is an extension of the Cross-Language Information Retrieval paradigm. Users who are unfamiliar with the language of documents retrieved are often unable to obtain relevant information from these documents. The objective of Cross-Language Information Access is to introduce additional post retrieval processing to enable users make sense of these retrieved documents. This additional processing may take the form of machine translation of snippets, summarization and subsequent translation of summaries and/or information extraction.

There have been efforts globally towards development of such systems. Cross-Language Evaluation Forum (CLEF), NTCIR Asian Language Retrieval, Question-answering Workshop and others have been working towards achieving the similar goals. In Indian context the need of such system becomes more evident that being multi-lingual country, the people here are familiar with more than one language. The availability of such system

helps in reaching the information if it is available in language other than the language of query. In order to meet this requirement, the CLIA (Cross Lingual Information Access) project has been initiated.

3 System Features

The system is intended to search different documents in Indian languages. Once the user starts the system, an initial screen with logo is displayed. By default, the screen is displayed in Hindi or English depending on the default language selected on the browser of the user. If the user wants to display this initial screen in any other language, he/she can select the language from the bottom of the screen. The screen is then displayed in the selected language. At present, the screen is available in seven languages: Hindi, English, Marathi, Punjabi, Bengali, Tamil and Telugu. To search a document, the first activity the user performs is the selection of the source language. Selection of source language allows the user to enter the text in the selected language.

•Selection of the Source Language: The user can select the source language by clicking a drop-down box. The system displays the languages available to select the source language.

•Entering String for Search: The user enters the query string on which the search is to be made in the appropriate place. The system allows the user to enter the string in the source language selected by the user using a soft keyboard for the language.

•Search the Web or the Site: Once the string is entered, the user should select whether to search the local site or the World Wide Web. The user can then click the search option to search the site for the string entered.

• Displaying the Results: Once the query is properly expanded and translated, it is used to search the web or the local site and the documents are retrieved according to the query. The snippets of the retrieved documents are displayed in the original language of the document as well as in the source language selected by the user. Thus, if the source language selected is Bengali, the user can enter query string in Bengali, the CLIA system searches for documents in English, Hindi and Bengali either from the web or the local site. The snippets of the retrieved documents are displayed in English/ Hindi and Bengali.

•Advanced Search: The user can also select the advanced search option and the CLIA system displays all the options accordingly. The user can select here the domain for which he/she wants to search the documents. At present, the tourism and health domains are available. The user can also select the number of items to be displayed on a single page. By default, the system displays 10 items on a single page. Once the selection is made, the user can click the 'search' option to start the search. In the advanced search option, the CLIA system provides summary as well as extracted information in the form of predefined information extraction templates, of the retrieved documents along with the generated snippet. The summary and the extracted information templates can be displayed in the original language of the document as well as in the source language selected by the user.

4 Technical Details

The CLIA system achieves its performance by means of the following five subsystems:

- Input processing
- Search
- Processing of Retrieved Documents
- Output Generation
- UNL Based Search

The main purpose of each of these subsystems is described below:

• Input Processing Subsystem

Input processing analyses the query entered by the user using language processing tools, expands the query to add more relevant terms and based on its analyses, either translates or transliterates all the query terms to the target language and then provides this as input to the search modules. The CLIA Input Processing subsystem consists of Language Analyzer (Tokenization, Named Entity Recognition, Multiword Expression, Stop word identification, Stemmer), Query Expansion (Pre- and Post-Translation Query Expansion), Query Trans-

lation and Query Transliteration. The CLIA Focus Crawler subsystem consists of Classifier, Language identifier and the Parallel crawler.

- **Search Subsystem**

The search subsystem lies at the heart of the CLIA-search engine. The main purpose of this module is to:

(a). Crawl the web and download files for a spe cific language and domain.

(b). Extract the text part in these documents and perform certain processing on those texts and convert them into indices.

(c). Extract results for a particular query by looking up the indices built so far.

(d). Arrange the document references returned by the search subsystem according to some order depending upon page ranking.

- **Document Processing Subsystem**

The document-processing module facilitates the access of documents written in English, Hindi and in the other languages. The documents crawled from the web are preprocessed using language processing tools for extracting information and translating the extracted information into target languages. This module consists of many language-processing tools such as Document Converter, Language Pre-processors, POS taggers, Text Chunker, Named Entity Recognizer, Domain Dictionaries, Information Extraction Engine and Translation engine. These modules are used in processing the documents.

- **Output Generation Subsystem**

This subsystem consists of the snippet generation, summary generation and the snippet translation modules. Brief details of the modules are described below:

(a). Snippet Generation: The Snippet Generation Module generates the snippet corresponding to the retrieved document. This module gets the parse text of the retrieved documents and the query from the search engine and generates the Snippet of each document and returns the generated snippet on the output screen.

(b). Summary Generation: The Summary Generation module generates the summary corresponding to the retrieved document. This module gets the parsed text of the retrieved documents and the query from the search engine and generates the summary of the documents.

(c). Snippet and Summary Translation: Generated snippets for English and Hindi documents are translated to the query language. If the query language is Hindi, then English documents are translated to Hindi. Translated snippet in the query language is displayed on the output screen along with the original snippet.

- **UNL-Based Search Subsystem**

The advanced search system uses UNL as a language independent intermediate representation to enable translation between the languages. The advanced search using UNL is based on concepts, and relations between concepts rather than bag of words. Hence it enables semantic search. Although the current system is designed for Tamil, it can be extended to other languages.

5 Future Roadmap

The functionalities of the CLIA system have been currently developed for Bengali, Marathi and Telugu. The search option has been limited to the crawled documents that are stored and indexed in the CLIA server. The crawled documents are in the tourism domain. At present, the user can provide 3-4 word queries to the CLIA system using soft keyboards for the respective language. The output of the system shows only the snippets in the original language of the document.

The CLIA system is being enhanced to provide full functionalities in the other Indian languages, i.e., Hindi, Tamil and Punjabi. The search option is expanded to provide search facility on the web also. Work is also going on for providing CLIA functionalities in the health domain. In future, snippet translation, summary generation and translation as well as information extraction templates generation and translation are going to be included in the CLIA system. The evaluation engine will judge the CLIA system based on the ranks of the relevant documents in the list of documents retrieved by the system.

Author Index